Pose Error Estimation of a Quadcopter in the Outdoors

by

Jacobus C. Lock

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Mechatronic) in the Faculty of Engineering at Stellenbosch University

Supervisor: Dr. WJ Smit
Co-supervisor: Mr. J Treurnicht

March 2016

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2016

Abstract

Pose Error Estimation of a Quadcopter in the Outdoors

JC Lock

Department of Mechanical and Mechatronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MEng (Mech)

March 2016

The quadcopter industry is a fast-growing and maturing industry which produces 'dumb' (or manually controlled) unmanned aerial vehicles (UAVs). They are also commonly equipped with controllers which make them able to autonomously carry out a flight mission without a human pilot. Industry is becoming increasingly interested in integrating quadcopters into their respective workforces in an attempt to automate some processes. However, national governments are worried that if left unregulated, UAV quadcopters may pose a safety and security threat to society, particularly if they are to work autonomously in the outdoors.

There are a number of improvements that can be made to increase the safety of quadcopters. This research project investigates how well quadcopters can estimate their position and orientation, or pose. If this is known, it can be integrated into a control model as an error term to improve the performance and safety rating of a quadcopter.

Indoor measurement tools cannot be used, since the quadcopters of interest rely on GPS data. Therefore, a computer vision-based pose measurement system (CVS) was investigated, designed and implemented to measure the pose of a quadcopter in the outdoors. The system's measurements were compared to a quadcopter's pose estimates recorded during an outdoor test flight.

The results show that the CVS's measurements are more accurate than the quadcopter's in all the dimensions except for the yaw. It was found that the quadcopter's position estimation error is approximately 150 mm for x, y and z, and 3.27° and 1.9° for the roll and pitch dimensions. These results can be used and integrated into a quadcopter's control model.

Uittreksel

Posisie en Oriëntasie Afskatting van 'n Vierbeen Helikopter in die Buitelug

JC Lock

Departement Meganiese en Megatroniese Ingenieurswese,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MIng (Meg)

Maart 2016

Die vierbeen onbemandevliegtuigbedryf is 'n snelgroeiende industrie wat vliegtuie produseer wat met die hand of heeltemal outonoom, sonder 'n menslike vlieënier se inset, 'n vlugopdrag kan voltooi. Industrie stel toenemend belang daarin om sulke onbemande vliegtuie in hul werksmag te integreer om prosesse tot 'n mate to outomeer. Wêreldsregerings is egter bekommerd dat indien die onbemandeveliegtuigbedryf ongereguleerd gelaat word, sulke vliegtuie 'n gevaar sal inhou vir die samelweing, veral as hulle gelaat word om outonoom in die buitelug te werk.

Daar is verskeie verbeterings wat gemaak kan word om die tegnologie se veiligheid te verhoog. Hierdie navorsingsprojek stel ondersoek in om vas te stel hoe akkuraat 'n onbemande vliegtuig sy posisie en oriëntasie kan afskat. Indien hierdie waardes bekend is, kan dit geïntegreer word in 'n beheermodel om so 'n vliegtuig se werksverrigting en veiligheid te verhoog.

Binnemuurse meetinstrumente kan nie gebruik word hier nie, aangesien die onbemande vliegtuie van hierdie projek staatmaak op hul GPS lesings. Dus, in hierdie projek is daar 'n rekenaarvisiestelsel ontwerp, getoets en geïmplimenteer om 'n vierbeen helikopter se posisie af te skat in die buitelug. Die vierbeen helikopter se afskattingsakkuraatheid was dan gevind deur sy afskatting te vergelyk met dié van die rekenaarvisiestelsel in 'n buitemuurse vlugtoets.

The resultate toon dat die rekenaarvisiestelsel se metings meer akkuraat is as die vierbeen helikopter s'n vir alle meetdimensies, buiten die afwykingshoek. Dit was gevind dat die helikopter se posisieafksattingsfout ongeveer 150 mm is in die x, y en z dimensies, terwyl die hoekafskattingsfout 3.27° en 1.9° is vir die rol- en hellingshoeke. Hierdie resultate kan geïmplimenteer word in 'n vierbeen helikopter se beheermodel.

Acknowledgements

Besides my supervisor, Dr. WJ Smit, I would like to thank STERG for the opportunity to study with them and the facilities they provided. Furthermore, I would like to thank CRSES for the funding they provided. I would also like to acknowledge the staff at Tygerberg's three-dimensional Motion Capture laboratory and the rest of the Stellenbosch University staff who had a hand in this project.

Dedications

I dedicate this thesis to my parents and brothers, whose unwavering love and support pushes me to be the best man I can possibly be. And to all my friends who tried their best to keep me from finishing my project: nice try.

Ek dra hierdie tesis op na my ouers, broers, Jay, en hulle onwrikbare liefde en ondersteuning, waarsonder ek nie sou wees waar ek vandag is nie. Christian, die een is vir jou.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Variables

f	Focal Length	[Pixels]
x	X-Coordinate	[m]
y	Y-Coordinate	[m]
z	Z-Coordinate	[m]
θ	Pitch Angle	[°]
ϕ	Roll Angle	[°]
ψ	Yaw Angle	[°]

Vectors and Matrices

C	Camera Matrix
N	Intrinsic Camera Parameter Matrix
P	Pose Matrix
$\boldsymbol{P}$	Pose Vector
R	Rotation Matrix
$\boldsymbol{T}$	Translation Vector
$\boldsymbol{x}$	Two-Dimensional Camera Coordinate Vector
$\boldsymbol{X}$	Three-Dimensional World Coordinate Vector
$\boldsymbol{\epsilon}$	Error Vector
Σ	Covariance Matrix

Subscripts

b	Board
c	Camera
i	Indexer
M	Number of samples in data set

Abbreviations

ANN	Artificial Neural Network
CSP	Concentrating Solar Power
CVS	Computer Vision System
FFNN	Feed-forward Neural Network
GPS	Global Positioning System
H_{inf}	H-infinity
IMU	Inertia Measurement Unit
LQR	Linear Quadratic Regulator
MPC	Model Predictive Controller
MSE	Mean Square Error
PID	Proportional Integral Derivative
PnP	Perspective n-Points
RANSAC	Random Sample Consensus
RBFNN	Radial Basis Function Neural Network
RNN	Recurrent Neural Network
STERG	Solar Thermal Energy Research Group
SU	Stellenbosch University
UAV	Unmanned Aerial Vehicle

Chapter 1

Introduction

1.1 Problem Statement

Remote controlled model aeroplanes have been in use for decades and have been a favourite among hobbyists for almost equally long. However, the past decade has arguably seen the greatest increase in computing power and efficiency since the silicon transistor was first invented (refer to the law popularised by Moore, 1965). Thanks to the growth in the mobile technology industries, computers have not only grown more powerful, but also smaller, lighter, cheaper and more power efficient. The increase in computing power and decrease in size and cost have made it possible to place small computers onto a model aeroplane with the aim of having it autonomously control, or semi-autonomously assist in controlling, a model aeroplane.

Coupled with the rise in mobile computing power, control theory has also reached a point where it has a better understanding of the unstable, underactuated plant, such as a Segway, and manages to stabilise and control such plants in some cases. One case of interest is a multirotor unmanned aerial vehicle (UAV) which resembles a model helicopter with multiple rotors. The so-called quadcopter configuration is of special interest for this research project.

Autonomous UAVs are typically fitted with a multitude of sensors that provide the control system with the UAV's orientation and localisation data. These sensors normally include an accelerometer, gyroscope and GPS, along with others such as a barometer, optic flow sensor, magnetometer, etc. In its semi-autonomous state, the control system uses these sensor readings to keep the UAV level and stable while following a human pilot's flight instructions. Optionally, the pilot can engage the UAV in *mission* mode where the UAV autonomously completes a flight mission selected by the pilot without the pilot controlling it. A mission consists of a set of GPS coordinate waypoints and altitudes for the UAV to follow. The pilot can also set the UAV to *loitering* mode where the UAV's control system will keep it stable and level while holding the yaw angle and position constant.

A consequence of the smaller and lighter sensors UAVs are typically equipped with is that they are often less accurate than their larger, more powerful counterparts. The implication this holds for UAVs in general can readily be observed from most UAVs in *loiter* mode where there is often significant drift around its set point. The accuracy of the individual sensors are often known or can be determined, but due to the mathematical filtering and fusion of the different sensor readings, as well as other operations that the control system may perform on the sensor data, it is difficult to determine the resulting accuracy of the position and orientation measurements made by the UAV's sensor suite. As a result, the pose (a vector describing an object's position and orientation) estimation accuracy of an outdoor UAV is not yet known. This research project attempts to find and implement an affordable and reliable method to determine the pose estimation error of a typical quadcopter UAV.

1.2 Project Motivation

For many years, UAVs have mostly been the playthings of hobbyists and the researchers studying them. In recent years, however, the improvements that have been made to UAVs, which include quad-, hexa- and octocopters, and their controllers have drawn the attention of large corporations, such as Amazon and DHL, have expressed interest in incorporating UAVs into their respective workforces.

Similarly, national governments have also noticed the increasing commercial and industrial potential that UAVs have, while also recognising the dangers that they pose to society if left unregulated. As such, many governments have moved to regulate and place restrictions on how and where UAVs may be used. The general trend of the regulations, as demonstrated by the regulations released by the South African Civil Aviation Authority (2015), is that UAVs may be manually flown by a pilot anywhere below an altitude of 120 m, 50 m away from people and buildings and 10 km away from any aerodrome. These regulations allow for autonomous UAV flight, provided the UAV remain within radio line of sight of a human operator that can manually take over control of the UAV at any time should something go wrong.

At the moment, UAVs are not as safe as they should be and can pose a serious health hazard to their surroundings and people if handled incorrectly. This, coupled with a general lack of hardware- and software-based collision avoidance capabilities, makes aviation authorities very cautious about allowing fully autonomous flight (i.e. out of radio line of sight). There are many safety improvements that can be made to UAVs, such as an improved control strategy, hardware improvements (such as rotor shrouds) and other fail-safe and collision avoidance systems.

Having accurate pose data of a UAV in flight available is crucial to implementing an effective control strategy and collision avoidance system. As

mentioned in Section 1.1, UAVs do not estimate their own pose very well and are prone to sensor drift over time. This allows for a 'bubble' of pose uncertainty to be drawn around a UAV in flight, where the true pose of a UAV will be somewhere within the bubble, but the true pose is indeterminable. If the bubble's volume can be determined, it will allow a UAV's control system to generate safer, more efficient flight paths, which will improve the UAVs performance and make them a more attractive and safe option for governments and industry. Furthermore, the measurement system can be used to evaluate and improve the flight performance of a UAV.

1.3 Potential Use Case

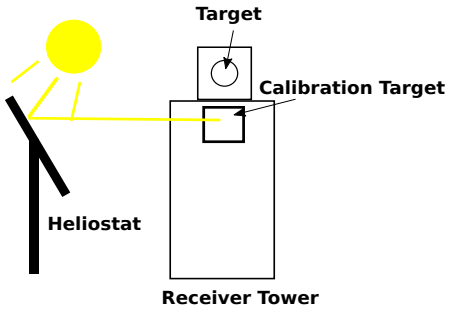
The results from this research project will be applicable to many different industries and applications. To illustrate this, an example of where it would be helpful to know how accurately a quadcopter can estimate its pose is presented and briefly discussed.

Concentrating solar power (CSP) is an attractive source of renewable energy which allows thermal energy to be stored for use at night time. However, a major hurdle to the construction of a CSP plant, in particular the central receiver configuration, is its hefty price tag. With CSP costing more per Watt than fossil fuels, as well as other renewable energy sources such as photovoltaic panels or wind turbines (IRENA, 2015), it is often difficult to convince financiers to invest in a central receiver CSP plant.

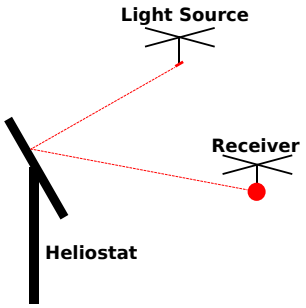
To prevent mirror deformation, wind loads, impacts from foreign objects and ground settling from affecting the heliostat's tracking path, each heliostat frame is designed to be heavy and sturdy and they are placed on strong foundations. They also come equipped with expensive actuators and gearboxes which minimise error build-ups during a heliostat's operation, contributing a significant portion to its total cost and Pitz-Paal (2005) finds that approximately 40 % of the initial capital expenditure of a CSP plant is spent on the thousands of heliostats placed around the receiver tower.

CSP as a concept is old and well documented, but the technologies used are still relatively new and widely researched. As such, CSP plant design has not yet reached full maturity in terms of efficiency and optimal design. Therefore, there are still ways to reduce the cost of CSP plants, such as improving the heliostat control scheme, optimising heliostat design and improving the thermal storage mechanisms, amongst others (IRENA, 2012).

Given the sheer size of a typical heliostat field, a small displacement in a mirror could influence the accuracy and effectiveness of that heliostat which is extremely undesirable given the lengthy calibration process required. Stellenbosch University (SU) and its Solar Thermal Energy Research Group (STERG) are attempting to make heliostats lighter, cheaper and reduce the need for expensive actuators and gearboxes, thereby allowing the heliostats to be placed



(a) Diagram of the current heliostat calibration procedure where the position of the reflected sun rays on the target are measured.



(b) A diagram of a proposed calibration procedure using quadcopters where the two quadcopter's positions are simultaneously measured.

Figure 1.1: A side-by-side comparison of how heliostats are currently calibrated and how heliostats can be calibrated by autonomous quadcopters.

on cheaper foundations with cheaper, less accurate actuators. This will lead to significant cost savings in a CSP plant's manufacturing stage. However, to achieve this without compromising the accuracy of the heliostats during operation, the lengthy calibration procedure currently in use may have to be optimised and possibly redesigned to allow the heliostats to be calibrated more often on a monthly, or even weekly, basis.

The current calibration procedure uses the sun as a light source and a section of the receiver tower as a reference target. This constrains the process to take place in day time and only allows one heliostat to be calibrated at a time. Since a heliostat field typically contains several thousand heliostats, this procedure may become very time-consuming. See Figure 1.1a for a diagrammatical example of the calibration procedure.

In this regard, STERG is investigating the possibility of using quadcopter UAVs to autonomously perform or assist with heliostat calibration by, for example, making one quadcopter the light source and another the receiver. This will allow as many heliostats to be calibrated simultaneously as there are pairs of quadcopters available. See Figure 1.1b for an illustration of the proposed calibration procedure. Since the heliostats must be calibrated with a fine degree of accuracy, this approach requires an estimate of the quadcopter's pose estimation error. Here, the pose estimation error refers to the difference between a quadcopter's on-board pose estimate and its real pose.

Since a quadcopter typically drifts around its set-point, some pose error term must be included into the heliostat calibration procedure. First the expected pose error, or 'bubble' of pose uncertainty, needs to be determined. After this is done, the pose error can be included into the calibration procedure and the procedure can be designed and refined around it.

Currently, the only pose measurements of a quadcopter available are its own on-board estimates produced by a combination of readings from its sensor suite, which typically includes a gyroscope, accelerometer and GPS. However, the on-board data cannot be used here, since its error has not yet been quantified. Consequently, another approach is required to determine a quadcopter's pose estimation error.

With a quadcopter's pose estimation error determined, it would be possible to autonomously calibrate a heliostat on a regular basis, which would lead to significant cost savings and allow CSP technology to become a more serious competitor in the renewable energy generation arena.

1.4 Existing and Proposed Solutions

The current state-of-the-art method in determining the pose of a UAV in flight is to use an indoor motion tracking system, such as a Vicon[1] system which uses a set of infrared cameras to track markers placed on an object. However, such systems cannot be used in this case since the UAVs of interest to this project require access to their GPS coordinates. The GPSs used in this project require a strong connection with a reasonably clear line of site to at least 6 different GPS satellites to produce accurate position data. These GPSs are also low-powered and small and therefore struggle to make good connections to satellites when indoors. Their line of sight to the satellites will also be affected, further reducing the GPS's accuracy. It was therefore decided that the quadcopters would be flown in the outdoors for this project. This implies that an outdoor pose measurement system is required.

Outdoor pose measurement systems, such as radar- or laser-based systems, can also be used to perform pose measurements of a UAV. However, these systems normally come at a premium cost. It was therefore decided to investigate and implement another pose measurement method that is cheap, accurate, repeatable and easy to use.

The proposed outdoor UAV pose measurement system is based on computer vision techniques where the pose data of an object can be extracted from image or video data containing the object. The system consists of a camera to capture the image data and a computer to perform the pose data extraction.

1.5 Project Objectives

The objectives of this research project are as follows.

- Design and implement a relatively cheap computer vision pose measurement system.

[1] www.vicon.com/

- Determine the measurement accuracy of the computer vision system.
- Use the computer vision system to determine the pose estimation accuracy of a demonstration quadcopter in flight.

1.6 Document Structure

This document begins with a review of existing literature and of previous research results in the fields relevant to this project, such as computer vision techniques, UAV control strategies and others. Afterwards, the design and implementation, as well as the determination of the accuracy of the computer vision pose estimation system is discussed. This is followed by a discussion on how the pose measurement error of the computer vision system was determined for subsequent measurements and used to determine the pose estimation accuracy of a quadcopter in flight. Finally, a conclusion on the significant findings and results, as well as shortcomings and potential improvements, are presented.

Chapter 2

Literature Review

2.1 Introduction

In this chapter, previous work and research findings relevant to this project are mentioned and discussed. These fields include the quadcopter platform, computer vision and machine learning techniques. Some concepts and terminology used in later chapters of this document are also explained and expanded upon.

2.2 Quadcopters

A quadcopter is an autonomous aerial vehicle (UAV) in a four-arm frame configuration. The frames typically come in an X or plus (+) shape with the control equipment and sensors located at the centre of the frame. Refer to Figure 2.1 for a picture of the Solar Thermal Energy Research Group's (STERG) Suncopter research platform which has an X configuration. They can also come equipped with either four or eight motors and props, though the four-rotor variant is commonly used and is used in this project.

Figure 2.1: A picture of the Suncopter used by STERG.

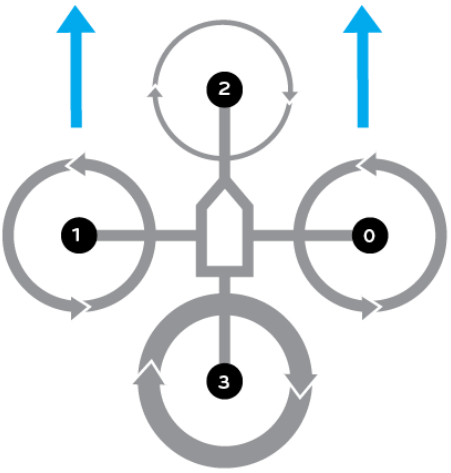


Figure 2.2: A diagram presenting the opposing axis rotation directions. The thickness of the arrows represent the power output of the motor (Majumder, 2014).

Quadcopters move freely in three-dimensional space and have six degrees of freedom: three in position space (x, y, z) and three in orientation space (θ, ϕ, ψ), where the orientation is defined by the Eulerian aeroplane angle scheme, i.e. roll, pitch and yaw angles. The rotors on each axis of the frame rotate in the same direction and each axis' rotors spin in opposite directions relative to one another. Refer to Figure 2.2 for a diagram demonstrating this.

Movement in each of the six dimensions is described below.

- Forward/backward or left/right movement: Keep one axis' rotor speed constant, while raising or lowering the speed of the rotors on the other axis (shown in Figure 2.2).
- Higher/lower: Increase or decrease all of the rotor speeds.
- Roll/pitch: Linear movement is achieved by tilting the quadcopter, therefore movement in the roll and pitch dimensions is achieved with the same process as described in the first point.
- Yaw: Increase or decrease the rotor speed of one axis relative to the other.

As described in Chapter 1, the goal of this project is to determine a quadcopter's pose estimation error, i.e. the difference between a quadcopter's true pose and its estimated pose. To be able to do this, it is important that the dynamics and control strategies of a typical quadcopter are well defined and understood. This section sets out to discuss the work that has gone into the

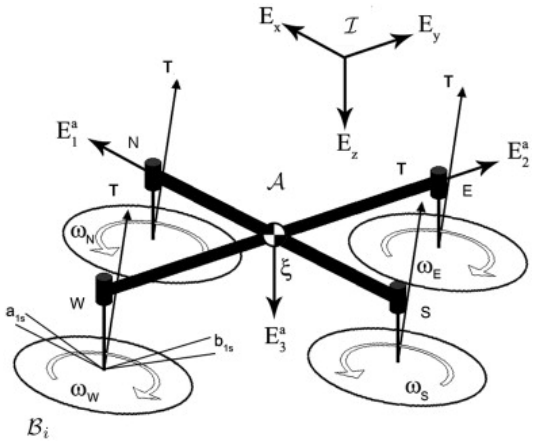


Figure 2.3: A diagram of the quadcopter model which includes the blade flapping dynamics discussed by Pounds *et al.* (2010).

dynamic modelling of a quadcopter, as well as the different control strategies that have been developed and implemented in the past.

2.2.1 Quadcopter Modelling

When designing a control system, arguably the most crucial part of the design process is to derive an accurate mathematical model of the plant, since an accurate plant model will lead to more stable response and superior performance. The plant in this case is a UAV in a quadcopter configuration.

As part of their X-4 Flyer project, Hamel *et al.* (2002) set out to derive a simple model for their plant using only rigid body dynamics and abstract force and torque actuators. As stated by Pounds *et al.* (2010), this model (like many others that were derived at the time) represents the quadcopter as a rigid body mass with inertia and autogyroscopics, which is only affected by gravity and actuator torques. Pounds *et al.* further argue that these simple quadcopter models do not accurately represent the complex helicopter-like behaviour exhibited by real quadcopters at high rotor speeds. The high-speed rotor effects include blade flapping effects, which affect the quadcopter frame's oscillatory modes, rotor flapping introduced by varying a quadcopter's yaw angle and variable airflow velocities over the rotor blades caused by changing roll and pitch angles.

In an attempt to create a more accurate quadcopter model that will allow a quadcopter to be more controllable at high rotor speeds, Pounds *et al.* set out to derive a model incorporating rigid body dynamics as well as the aerodynamic effects mentioned earlier. Their model is based, in part, on the diagram given in Figure 2.3.

Their resulting model was used to develop a simple Proportional Integral Differential (PID) attitude and altitude controller for the purpose of model verification. Their results show that the quadcopter stabilised itself (in indoor

flight) with a level of precision of ±1°, and ±5° during outdoor flight. The lower precision during outdoor flight is due to the added wind disturbances and sensor drift. They therefore conclude that their model is sufficiently accurate to safely control a quadcopter for hovering. However, they did not compare their more complex model against the model of Hamel *et al.*, which is based solely on rigid body dynamics.

2.2.2 Control Strategies

After an accurate model has been established, the next important part of designing a good control system is the controller itself. Many controllers have been implemented and tested on quadcopters over the years and different control strategies have also been investigated. Some of the most prevalent control strategies and controllers are discussed here.

Indoor vs. Outdoor Control

There are different types of quadcopters, with each of them equipped with different sensors and equipment. However, two types of quadcopters with distinctly different sensor and control approaches are relevant to this project. These are indoor and outdoor quadcopters and each of them work in different ways and implement very different control schemes.

To help with stability control, indoor quadcopters may or may not come equipped with an on-board inertial measurement unit (IMU), which typically includes an accelerometer and a gyroscope. However, they commonly solely rely on an external motion detection system which tracks the quadcopter's movement and provides position and orientation feedback to the controller, thereby closing the control loop. These quadcopters can be very accurate due to the highly accurate external motion sensing equipment which are capable of sub-millimetre levels of accuracy (Richards, 1999). As a result they are capable of performing remarkable acrobatic feats. However, these systems are restricted to carefully regulated and controlled indoor environments.

Outdoor quadcopters do not have the luxury of highly accurate external motion tracking systems and have to rely on their on-board sensors to provide the controller with pose feedback. A quadcopter's on-board sensor suite may vary between quadcopter platforms; however, they almost certainly come equipped with an IMU to provide pose data, but since the IMU readings for position data drifts with time due to integration errors, a GPS sensor is added to provide a base-line reading of a quadcopter's position. Other sensors that may be included are magnetometers, barometers, visual feedback sensors and sonar sensors. To combine the readings of the different sensors a filtering technique, such as an Extended Kalmann Filter, can be used. The pose estimation error of the combination of the different readings are, in theory, less than the

most accurate sensor in the suite, but this has not yet been proven and the exact error margin is yet to be determined.

Hover Control

Hover control refers to a quadcopter's ability to hover and remain stable at a set point in three-dimensional space. The stable hovering of a quadcopter has been the focus of many projects and research papers in the past 15 years. As a result, many different control methods and schemes have been investigated, implemented and compared.

Bouabdallah *et al.* (2004), as part of their OS4 project, compared the modern linear quadratic regulator (LQR) and classic PID controller, with respect to the control performance (disturbance rejection, reference tracking, etc.) of a quadcopter.

They found that the PID controller produced better results than the LQR in terms of reference tracking and dynamic performance. This is a surprising result, since LQR controllers normally excel at controlling an unstable, underactuated plant such as a quadcopter platform. They suspect that the reason for this surprising result is that they neglected the effects of actuator dynamics, such as blade and rotor flapping, in their quadcopter model and the PID was better at handling plant uncertainties. They expect that an LQR controller will outperform a PID controller if a more accurate model is used which takes aerodynamic effects into account.

Some researchers have also investigated controlling a quadcopter using an H-infinity (H_{inf}) control structure and a model predictive controller (MPC). Most notably, Raffo *et al.* (2010) have done extensive research on this topic. MPCs are modern controllers that drive a plant's state to a reference state within predefined constraints (eg. motor saturation, model dynamics, etc.), while a properly designed, non-linear H_{inf} controller is very good at rejecting disturbances (eg. wind gusts, motor vibrations, etc.) and are robust to model uncertainties. They opted to combine the two controllers in an intelligent manner to extract the best performance from their quadcopter.

In their simulations, they found that the resulting controller exhibited good performance characteristics; presenting good reference tracking, proving to be robust with uncertain mass and inertia terms and deals well with disturbances on all six degrees of freedom at different points in time. However, they are yet to implement their controller configuration on a real quadcopter. Although the algorithms and methods they used are computationally efficient, it may still prove to be too computationally intensive for the limited computing power on-board a UAV. Given the fast growth of processing power, however, this controller configuration may become a more viable option in the near future.

Controllers for enabling a quadcopter to hover have already been successfully designed and implemented, and it is therefore possible to stabilise and accurately control a quadcopter for hovering operations.

2.3 Computer Vision

Computer vision is a diverse field which primarily focuses on finding methods for acquiring, processing, analysing and understanding images captured of the world. The 'world' in this context refers to the three-dimensional world as perceived by living beings. There are many sub-fields of research, but a common theme across all of them is to make a computer mimic the human ability to perceive and understand an image and elicit an appropriate response to different visual inputs.

The fields of interest for this project are feature detection, feature tracking and pose estimation. This section discusses the work that has been performed in these fields of research.

2.3.1 Camera Matrix

A digital image is a collection of two-dimensional colour intensity vectors representing a collection of three-dimensional space vectors. These collections of vectors are related by a matrix, C, known as the camera matrix. The camera matrix contains the intrinsic parameters of the camera that recorded the image, i.e. the focal lengths and principle point of the camera, as well as the extrinsic, or pose, parameters of the camera, i.e. the camera's position and orientation information. The camera matrix, C, as derived by Heikkila and Silvén (1997), is

$$C = NP. \tag{2.1}$$

The matrix N in Equation 2.1 is given by

$$N = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.2}$$

Here, f_x and f_y describe the focal lengths of the camera, while u_0 and v_0 represent the camera's principal point coordinates. The principal point, also known as the focal point, is where the camera's axis crosses the image plane and is ideally situated in the centre of the camera lens. However, due to manufacturing defects, this is rarely the case. Refer to Figure 2.4 to see what the focal length and principle point coordinates represent.

The pose matrix, P, from Equation 2.1 is given by

$$P = \left[R|\boldsymbol{T}\right] = \begin{bmatrix} r_{11} & r_{21} & r_{31} & t_1 \\ r_{21} & r_{22} & r_{32} & t_2 \\ r_{31} & r_{23} & r_{33} & t_3 \end{bmatrix}. \tag{2.3}$$

In the matrix P, R is a 3×3 cell matrix describing the orientation of the camera and $\boldsymbol{T}$ is a three-dimensional vector describing the position of the camera. The pose information in P is given relative to some reference object or plane.

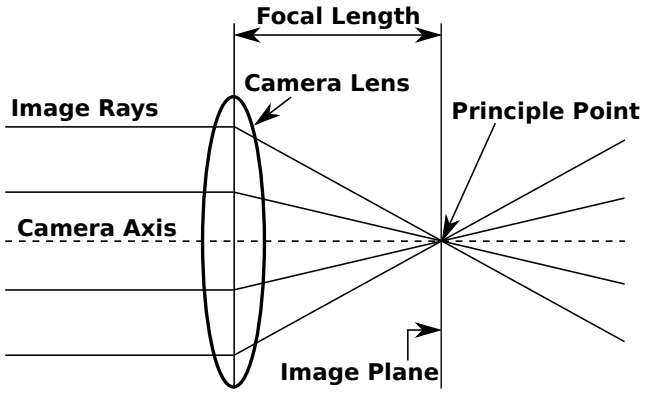


Figure 2.4: Diagram showing what the focal length and principle point parameters represent.

The two-dimensional image projection of an object in three-dimensional world space is related through C with the relation given in Equation 2.4. The camera matrix is commonly determined through some camera calibration procedure. One such a procedure is discussed in Section 2.3.2.

$$\boldsymbol{x}_c = C\boldsymbol{X}_w \tag{2.4}$$

In Equation 2.4, $\boldsymbol{x}_c$ is a homogeneous image vector $[x\ y\ 1]$ and $\boldsymbol{X}_w$ is a homogeneous world coordinate vector $[X\ Y\ Z\ 1]$.

2.3.2 Camera Calibration

A properly calibrated camera is an important part of any computer vision system, since the accuracy of the data extracted from an image strongly depends on the accuracy of the calibration procedure and results. The goal of the calibration procedure is to produce the matrix C (as given in Equation 2.1), as well as finding the camera's distortion coefficients introduced by low-quality or fish-eye lenses. There are various camera calibration procedures available: from the two-step calibration described by Melen (1994) to the classical approach given by Slama *et al.* (1980) where a non-linear error function is minimised. However, the minimisation problem presented by Slama *et al.* is computationally inefficient and slow, while Melen's method does not account for image distortion and correction. A popular calibration method is the four-step method proposed by Heikkila and Silvén (1997) as an extension to the two-step method which was the prevalent calibration procedure at the time.

The *calibrateCamera()* function of the OpenCV[1] computer vision library (Bradski, 2000) makes use of the four-step method. The details of this method is beyond the scope of this research project, however, a broad overview of the steps and equipment required to calibrate a camera is provided.

[1]Open-source computer vision library, OpenCV v2.4.8. Source-code available at github.com/Itseez/opencv

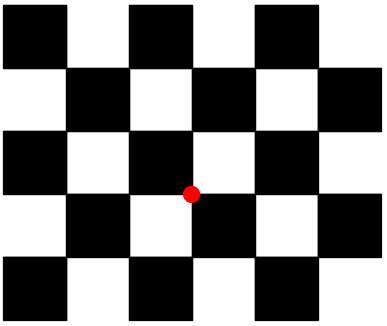


Figure 2.5: An example of a typical chessboard pattern used for calibration.

To perform the calibration and find the camera matrix, OpenCV requires two sets of data: one two-dimensional image data set, $\boldsymbol{x}_c$, as well as a set of corresponding three-dimensional data points, $\boldsymbol{X}_w$. This implies that image data of an object, where the dimensions and coordinates of certain features are known, must be recorded. In practice, any well-characterised object can be used for calibration. For example, some calibration methods rely on a three-dimensional cube covered in precisely laid out markers. However, since manufacturing and distributing such precisely constructed objects to a large audience is infeasible, OpenCV opts to use a more convenient flat, regular pattern, such as chessboard or asymmetrical dot pattern. Figure 2.5 shows an example of a typical chessboard pattern generated by OpenCV. With this flat pattern, the features used to populate the data sets would be the corners on the chessboard, i.e. where one black block meets another black block. The drawback to this approach, however, is that multiple views of the flat calibration pattern are required, whereas a single image of a three-dimensional object would suffice. However, more data points would allow the optimisation procedure built into the algorithm to find a more accurate result.

In the case of the chessboard pattern, acquiring the two-dimensional pixel coordinates of the corners is accomplished by using OpenCV's *findChessboardCorners()* and *findCornerSubPix()* functions, which make use of the corner detection algorithm described by Harris and Stephens (1988). The three-dimensional world coordinates of the features are fed into the calibration function according to an axis-system and measurement unit defined by the programmer. These coordinates can be as simple as a vector containing the feature coordinates in square units, where, for example, the corner from Figure 2.5 will be represented as $(3, 2, 0, 1)$ in homogeneous coordinates (note that z will be zero since the board is flat).

For the best calibration results, OpenCV recommends that camera calibration takes place within a well-lit room with a white background, using a calibration pattern with a wide white border in clear view of the camera and at different positions and orientations relative to the camera. These recommen-

dations are mainly to increase the contrast between the black features and the white background on the calibration pattern and make it easier for OpenCV to accurately find each feature's pixel coordinates. Furthermore, the more diverse the position and orientation data is, the more accurate the estimate for the intrinsic camera parameters will be.

Once the two-dimensional and three-dimensional data sets are known, OpenCV's *calibrateCamera()* function can determine the camera matrix, C.

2.3.3 Perspective n-Points Problem

The Perspective n-Points (PnP) problem, as stated by Horaud *et al.* (1989), 'is the problem of finding the position and orientation of a camera with respect to a scene object from n correspondence points', where the scene object would normally be a well-characterised calibration object or pattern. It is a well-researched sub-field of computer vision with different solutions to the problem that have been proposed. They include some non-iterative solutions, such as the P3P solution proposed by Gao *et al.* (2003) and the PnP solution by Lepetit *et al.* (2009) and Schweighofer and Pinz (2006), and iterative solutions such as the method proposed by Lu *et al.* (2000).

It was found that iterative methods produce very accurate results, but can become unstable if it is not properly initialised and can take a long time to converge. Conversely, the non-iterative ePnP method by Lepetit *et al.* implements Schweighofer and Pinz's robust solver and produces results whose accuracy is comparable to those produced by its iterative counterpart. However, it produces these results in a fraction of the time, having a big-$\mathcal{O}$ complexity that grows linearly ($\mathcal{O}(n)$), as opposed to competing non-iterative methods which commonly have a big-$\mathcal{O}$ complexity in the order of 4 or more ($\mathcal{O}(n^4)$). Unfortunately, the accuracy of the ePnP method's results are fairly dependant on the number of sample points, i.e. the number of feature correspondences between the three-dimensional features and their two-dimensional projections.

The OpenCV library has implementations of both the P3P and ePnP methods, as well as its own solution based on Levenberg-Marquardt optimisation, as described by Levenberg (1944) and Marquardt (1963), where the pose that minimises the reprojection error - that is the sum of the squared distances between the actual two-dimensional points and the projected two-dimensional points - is determined and selected (OpenCV, 2015).

OpenCV's *solvePnP()* function can be used to determine the pose of a camera relative to a calibration pattern. This can be accomplished as follows. During the camera calibration procedure, the intrinsic parameter matrix, N, of Equation 2.2 is determined. Then, using a calibration pattern, a set of three-dimensional feature coordinates and their corresponding two-dimensional projection is obtained. Following from Equation 2.4, with the matrix C, the two-dimensional image projection vector, $\boldsymbol{x}_c$, and the three-dimensional object coordinate vector, $\boldsymbol{X}_w$, the pose matrix, P, can be found.

This matrix contains the position and orientation data for the camera relative to the calibration pattern.

2.3.4 Random Sample Consensus

For both the camera calibration and PnP solving functions, two-dimensional image projection data of a three-dimensional object is required. As mentioned, OpenCV's *findChessboardCorners()* function can automatically detect corner features on a chessboard pattern and provide the two-dimensional projection data. However, the methods employed are prone to erroneously classifying some features as corners, introducing unwanted noise into the system.

To remedy this, Fischler and Bolles (1981) proposed a new algorithm to iterate through a data set and reject any outlier data. This algorithm is known as RAndom SAmple Consensus (RANSAC) and is commonly used in the computer vision field to determine if an image feature, e.g. a corner on a chessboard, has been classified correctly and rejecting it if it was not, thereby reducing the amount of noise in a data set.

2.4 Machine Learning

Machine learning is a research field in computer science with strong ties to the fields of mathematical statistics and optimisation. The field is well-established and has its roots with a paper by Turing (1950) in which he poses the question, 'Can machines think?'. Michalski *et al.* (2013) offers a somewhat more formal definition: 'A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E'. This means that researchers in the machine learning field are attempting to find efficient methods and algorithms that will allow a computer to be trained to make intelligent decisions when presented with an arbitrary input data set.

An everyday example of a system that uses machine learning is the face detection software that often come packaged with digital cameras. Here, the camera has been trained to search for facial information within the image, and consequently detects them when presented with other images containing faces.

Various machine learning algorithms and types have been developed, each of them having their unique advantages and limitations. Here, brief discussions on the most prevalent machine learning methods are provided.

2.4.1 Artificial Neural Networks

Artificial neural networks (ANNs) are a family of machine learning algorithms which have seen a rise in popularity in recent years. The idea behind ANNs is to model the vast network of interconnected neurons in a biological brain in

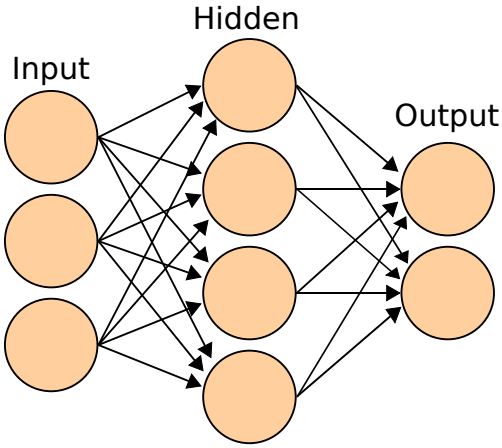


Figure 2.6: Diagram of a simple ANN (Wikimedia Commons, 2006).

such a way that the network can be trained to recognise patterns and make decisions based on what it perceives, much like any animal or human would.

Normally, an ANN consists of a multitude of artificial 'neurons', or nodes, numbering anything between a handful to many millions, arranged in a series of layers. Each of the nodes in every layer is connected to each other node in the layers on either side of it, forming a vast network of interconnected nodes forming something analogous to a living brain. A diagram of the layout of a simple feed-forward ANN is given in Figure 2.6.

Each network has two special layers called the input and output layers. The input layer accepts information from which the ANN's designer wants data extracted. The output layer is responsible for producing the output which contains the information on how the network responded to the input excitation data. In-between these extreme layers lie the so-called 'hidden' layers. These layers form the majority of the network and are responsible for interpreting the input data and producing the network's output.

The connections between the hidden nodes are represented by a weighting factor which are determined during a training process. These weights define how much influence the nodes have over another, i.e. if a weight is positive, it excites another node, whereas if it is negative, it suppresses the node. The input information traverses the hidden layers, activating the next node with the highest connection weighting. The output data is then determined by which of the nodes were traversed in the hidden layers. The connections between the hidden node layers are mostly responsible for the output and different connection schemes provide different performance characteristics.

Consider a simple example: you wish to create an ANN to recognise whether a picture contains a man or a dog. You train it with 25 images of different men and dogs, telling the ANN which picture contains which. After it has been trained, you show it a picture containing a young boy which is totally unfamiliar to the network. Based on the way you trained it, the net-

work should recognise enough human and male features in the child to come to the conclusion that it is more likely that the child is a man rather than it is being a dog.

This ability to classify information that technically falls out of the network's training environment is one of the strengths of ANNs. This advantage, as stated by Tu (1996), is ANNs' ability to implicitly detect any non-linear relationships between multi-dimensional input and output dimensions. They can also be trained using different training schemes and some modern software packages and libraries have also made it fairly easy to develop a model without any formal statistical training. Some drawbacks of ANNs are that they may require an immense amount of computing power if many nodes are initialised, they are prone to overfitting data when a poorly selected number of nodes are used and the trained models are extremely 'black-box' solutions, making it difficult to identify and characterise the relationships between the nodes.

Different ANN topologies and layouts, have been developed and proposed. Some of them are briefly discussed in the following sections.

Feed-forward Network

The oldest, and arguably the simplest ANN topography is the feed-forward network (FFN), also referred to as multi-layer perceptron if there are multiple layers in the hidden layer. Its layout is typically similar to the one given in Figure 2.6, with a single input and output layer, as well as a single or multiple hidden node layers.

The FFN uses some form of supervised training algorithm, with the back-propagation algorithm commonly used. Here, the hidden nodes' weights are adjusted until the output the network produces is as close as possible to the target output specified by the designer. This configuration's biggest attractions are its simplicity, relatively fast testing speed, depending on the number of hidden layers, and its ability to derive non-linear relationships between dimensions. However, FFN's are prone to converging very slowly and sometimes getting stuck in local minima during the training phase, as stated by Svozil *et al.* (1997). However, improvements to the backpropagation training scheme have reduced this effect.

Recurrent Neural Network

The recurrent neural network (RNN) is a family of neural networks. The hidden nodes of an RNN make provision for feedback between the different hidden layers and the output layer, creating an internal state for the network. See Figure 2.7 for a diagram of a RNN topography.

In contrast to the FFN, this internal saved state allows the RNN to process arbitrary sequences of input data, making it adept at processing unsegmented speech or handwriting patterns. However, the added complexity of adding

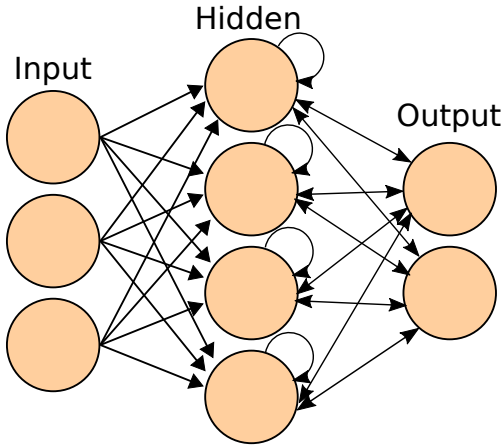


Figure 2.7: A diagram of a simple RNN (adapted from Wikimedia Commons, 2006).

feedback loops between the different layers becomes computationally expensive, especially when large networks with many layers and inputs are used. Increases in computing power and improvements in the training process, as well as a better understanding of RNNs and ANNs in general, have alleviated the computational expense somewhat.

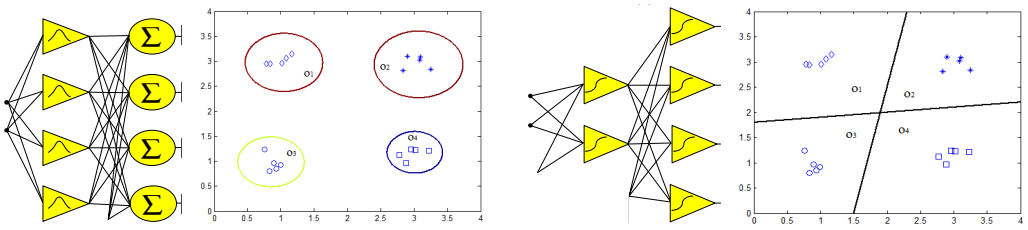
There are different RNN configurations available. These include the Hopfield network (Hopfield, 1982), the echo network (Jaeger, 2001) and the recurrent multilayer perceptron network (Tutschku, 1995).

Radial Basis Function Network

The radial basis function neural network (RBFNN) is another type of neural network and is a sub-family of the RNN family, as stated by Wilamowski and Jaeger (1996).

The RBFNN topology is fixed to a three-layer architecture, with one input layer, one hidden layer and one output layer. The input layer provides the input. The hidden layer then remaps these inputs to make them linearly separable, where the output layer does the separation and outputs the data (Xie *et al.*, 2011).

Despite belonging to the same family of ANN, there are a number of significant differences between the RBFNN and RNN. Firstly, the three-layer RBFNNs are simpler than multi-layered RNNs, making the training process for RBFNNs generally faster than for an RNN. Secondly, and most importantly, as stated by Xie *et al.*, is the difference in how the RBFNN and RNN classifies the data: the RBFNN data clusters are separated by a hyper sphere, whereas RNNs use arbitrarily shaped hyper surfaces. See Figure 2.8 for a visualisation of these class separation strategies. This makes RBFNNs an attractive option to interpolate multidimensional data.



(a) (b)

Figure 2.8: A comparison between an RBF classifier in Figure 2.8a and an RNN classifier in Figure 2.8b as shown by Xie *et al.* (2011).

Xie *et al.* further determined that RBFNNs are well suited to interpolate noisy data where the data surfaces contain regular valleys and peaks. In contrast, normal ANNs and RNNs are more efficient for classification problems and for well-conditioned, regularly spaced data.

As described by Skala (2012), the function on each node is given by

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{M} \lambda_j \phi(||\boldsymbol{x}_i - \boldsymbol{x}_j||). \tag{2.5}$$

Therein, λ_j is the node weighting factor and $\boldsymbol{x}_j$ are the kernel centres which are determined during the model training phase. The function ϕ is the radial function of the euclidian norm of the distance between the node centre and the input vector. This radial function is variable and the designer can select the function which best describes the data, although a Gaussian radial function, i.e. $\phi(\boldsymbol{r}_{ij}) = e^{-(\frac{\boldsymbol{r}_{ij}}{\epsilon})^2}$, $\boldsymbol{r}_{ij} = ||\boldsymbol{x}_i - \boldsymbol{x}_j||$, is commonly used.

2.4.2 Support Vector Machines

Support vector machines (SVM) is a widely used classification technique first proposed by Cortes and Vapnik (1995). It works by finding a hyperplane between two data classes that separates the classes by the widest possible average margin. See Figure 2.9 for an illustration of this separation.

Cortes and Vapnik's original method was limited to linear hyperplanes. Since then, the algorithm has extended to non-linear hyperplanes by applying what is known as the 'kernel trick', as described by Amari and Wu (1999). An example of such a non-linear separator can be seen in Figure 2.10.

SVM's are limited to binary problems with two classes, somewhat limiting their potential for high-dimensional problems. Regardless, they are extremely popular in scientific circles due to their accuracy and relative simplicity.

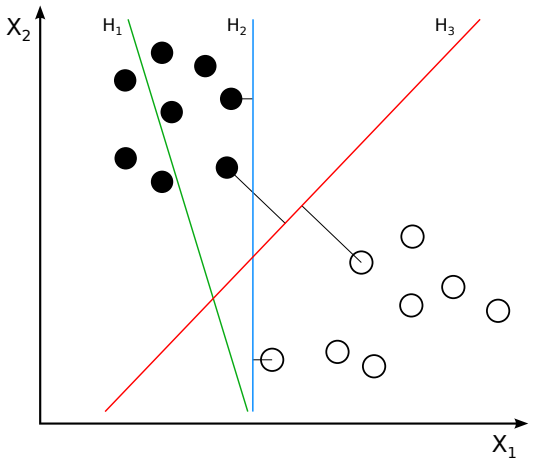


Figure 2.9: An SVM plot illustrating different class separators (Wikimedia Commons, 2012). H_1 does not separate the classes, while H_2 has only minimal class separation. H_3 exhibits the widest average separation margin.

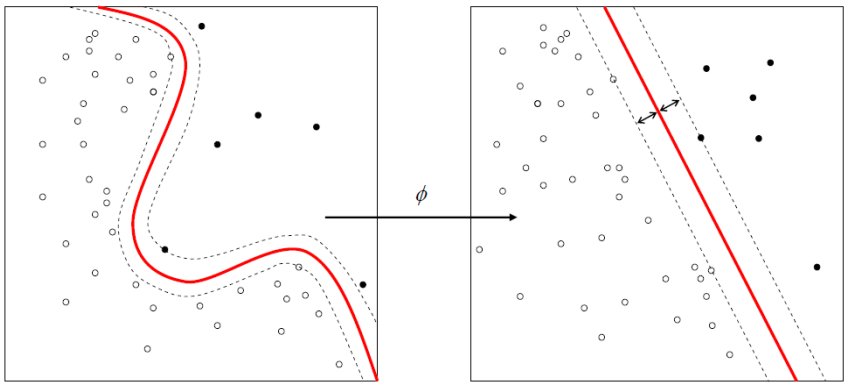


Figure 2.10: An example of an SVM with a non-linear kernel separator (Wikimedia Commons, 2011).

2.5 Conclusion

In this chapter, several concepts relevant to this project have been mentioned and discussed. Furthermore, past research findings, algorithms and techniques have also been identified.

It was found that the stable control of a quadcopter in the outdoors is possible, thanks to improved mathematical models and sensors. It has also been established that a computer vision pose measurement system is feasible and there are freely availible libraries which are capable of it. Lastly, it was determined that artificial neural networks are well suited to this project's high dimensional nature.

Chapter 3

Pose Measurement System Design

3.1 Introduction

The goal of this project is to determine the pose estimation accuracy of a quadcopter flying in the outdoors. In this context, the pose is a six-dimensional position and orientation vector.

In order to determine a quadcopter's pose estimation accuracy, two sets of data are required: the first is the quadcopter's own pose estimate and the second its true pose, as measured by an accurate external measurement tool. There are accurate measurement tools available for the outdoor environment, which include laser and sonar systems, however, these are fairly expensive systems. It was therefore decided to investigate an alternative method of measuring a quadcopter's true pose in the outdoors.

A computer vision-based system was investigated and later implemented. Using a computer vision system (CVS) for measurement is an attractive option due to it being simple, cheap, compact and easy to operate. However, the measurement accuracy of these systems differ between platforms. Therefore, a method of reliably determining a CVS's pose measurement accuracy must also be developed before it can be used to make pose measurements of a quadcopter.

This chapter sets out to provide detail on the design process of the CVS. First, the design and layout of the CVS is discussed. Then, the method employed to determine the pose measurement accuracy is provided. Finally the results of the entire process, from testing to data processing, are discussed.

3.2 System Layout

The CVS has both hardware and software components, both of which are important to making accurate pose measurements. In this section, the layout and design of the CVS, including its hardware and software components, are discussed, providing a broad overview of the system's make-up and how it functions.

3.2.1 System Overview and Requirements

The CVS is meant to measure the pose of a quadcopter in flight. It does this by using a calibrated camera to track a calibration board stuck to the object whose pose is to be measured (the quadcopter in this case). It then follows the next steps to determine the object's pose.

Step 1 Detect and track the feature data of the calibration board, e.g. corners or dots on a flat board.

Step 2 Extract two-dimensional pixel coordinates of the features.

Step 3 Use a Perspective n-Points solver with the two-dimensional coordinates to determine the six-dimensional object's pose relative to the camera.

These steps are explained in more detail in this section.

The CVS's pose measurements must be accurate enough that it can be used as reference pose data when comparing it to a quadcopter's on-board pose data, thereby determining the pose accuracy of a quadcopter. The CVS has to meet the following requirements:

- The CVS must be able to make six-dimensional pose measurements.
- The CVS should make measurements with an accuracy comparable to the quadcopter's on-board estimate.
- The CVS should make reliable and consistent pose measurements.

A CVS was designed and implemented to meet these requirements. The proposed computer vision measurement system makes use of the following processes and procedures:

- Camera parameter estimation and calibration.
- Automated two-dimensional image feature detection.
- Pose data extraction and derivation.
- Outlier data elimination and reduction.

3.2.2 Software

The CVS is a system which extracts pose information from an arbitrary image. There is a strong software aspect to the CVS since it relies on well-researched and understood computer vision techniques and algorithms. To this end, the popular OpenCV library was used to perform the computer vision tasks. This library was used since it is free, readily available, has a wide online support

network and comes pre-packaged with a large variety of up-to-date and powerful computer vision functions. To perform the numerical matrix operations, the open source NumPy[1] library for Python[2] was used.

OpenCV was used to calibrate the CVS's camera, extract object feature data from video data and determine the pose of a calibration pattern by solving the perspective n-points (PnP) problem. The entire process is described next.

First, the camera was calibrated by following the procedure discussed in Chapter 2. Here, an ISO A1[3] 5×6 square chessboard pattern was used for calibration. This size and number of blocks were selected to allow the squares to be as large as possible to make it easier for the CVS to detect the corners when the board is held approximately 1.5 m away while allowing a wide white border to be drawn around the squares, as per OpenCV's recommendations.

Next, the feature data is extracted from the images containing the calibration board. To summarise the procedure, the OpenCV functions *findChessboardCorners()* and *findCornersSubPix()* are used to detect and extract two-dimensional coordinate data of the corners on the chessboard pattern from a still image. These coordinates, along with their three-dimensional world coordinates, are fed to the *calibrateCamera()* function to produce the camera matrix. The world coordinates are predefined by the programmer and contain the real world location of the corner coordinates (see Figure 2.5 for a more detailed explanation). The calibration procedure produces the camera matrix, C, as presented in Equation 2.1 and repeated in Equation 3.1 for convenience.

$$C = NP \tag{3.1}$$

In Equation 3.1, the matrix N is given by Equation 3.2 and matrix P by Equation 3.3.

$$N = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$P = \left[R|\boldsymbol{T}\right] = \begin{bmatrix} r_{11} & r_{21} & r_{31} & t_1 \\ r_{21} & r_{22} & r_{32} & t_2 \\ r_{31} & r_{23} & r_{33} & t_3 \end{bmatrix} \tag{3.3}$$

After the camera has been calibrated, C is known and can be used by OpenCV's PnP problem solver to extract the camera's pose data relative to the calibration board. This procedure is based on the relation given in Equation 3.4 which relates the three-dimensional world coordinate vector, $\boldsymbol{X}_w$, to its two-dimensional image projection, $\boldsymbol{x}_c$.

$$\boldsymbol{x}_c = C\boldsymbol{X}_w \tag{3.4}$$

[1] NumPy numerical mathematics library v1.8.2
[2] Python computer programming language v2.7
[3] Standard paper size 841 mm × 594 mm

Using the relation in Equation 3.4 and C's intrinsic parameters, OpenCV's *solvePnPRansac()* function can be employed to extract the pose matrix, P, of the camera relative to the calibration board, as well as reject any outlier points with the RANSAC algorithm. The PnP solver used is the robust PnP solver, employing Leverberg-Marquart optimsisation as described by Schweighofer and Pinz (2006). It was found that a 5×6 chessboard does not have enough corner features to guarantee accurate results from the ePnP method of Lepetit *et al.* (2009).

3.2.3 Hardware

The hardware requirements for the CVS are minimal and its setup equally simple. To allow the software to make six-dimensional pose estimations, the hardware needs to capture image data at an appropriate resolution for OpenCV to be able to detect and extract two-dimensional feature coordinate data. The camera should preferably not have a lens zoom function in order to keep the focal length constant throughout a measurement session. A computer running the OpenCV and data-processing scripts, as well as a calibration board, are also required.

The camera that was used is a single Microsoft LifeCamHD webcam capable of capturing 720p high definition video data at a rate of 30 frames per second. This camera has a variable zoom feature, however, which was controlled and set to a constant value by using the *uvcdynctrl* webcam control library[4]. A stereo camera setup could also have been used, however, the accuracy between the single and stereo camera in most respects is rather negligible. However, stereo cameras outperform a single camera in the depth dimension if only local image feature data is available, as is the case here (Saxena *et al.*, 2008). For this implementation, the single camera variant was used even though relatively bad depth estimates were expected. The motivation behind this choice is that a single camera system is simpler than a stereo system and the limitations of the system are also clear. In the future, the project can be expanded to a stereo camera system if it is found that the inaccurate depth estimate significantly affects the CVS's measurement accuracy.

The calibration board used was a flat, ISO A1-sized, 5×6-square chessboard pattern calibration board. A large board was selected to allow the squares on it to be large, affording the camera a good view of the corners and improving the data extraction performance. The large board size also allowed for a fairly wide white border around the squares, as per OpenCV's recommendation.

A laptop running Linux Mint 17.1 'Rebecca' was used as a ground control station for the quadcopter in subsequent measurement tests. In addition, it was used as a recording device along with the webcam and performed some of

[4] Webcam control library for the Video4Linux driver uvcdynctrl v0.2.4

the data processing tasks, although a more powerful desktop PC was used to perform the image processing tasks.

3.3 Measurement Test Design

Before the CVS could be used to measure the pose of a quadcopter in flight, the accuracy of its measurements was first determined. The PnP solving algorithm is, at its core, an optimisation problem and only produces pose estimates. In light of this, determining the accuracy of the CVS's pose measurements, or estimates, is an important step in the system's design phase.

To determine the measurement accuracy of the CVS, a measurement test was performed in an indoor environment where another external measurement device, whose accuracy is high enough that its pose measurements can be taken as ground-truth values, could record pose data. The CVS's measurement error could then be determined by comparing the CVS's measurements with that of the external measurement system's. Both systems were set to record the pose of a flat chessboard pattern that was moved and orientated by hand.

This section describes the test layout, including the external measurement device and its details, as well as the CVS's details for the test. Next, the measurement procedure is presented, followed by the steps taken to process the data during the post-processing phase.

3.3.1 Test Layout

This section describes the test layout, including the layout for the CVS and Vicon systems.

External Measurement Device Layout

The external measurement system used to record the ground-truth pose data is a Vicon indoor motion capture system. It is a widely-used commercial system with applications in the film, medical and sporting industries and its measurements can reach sub-millimetre accuracy, as found by Windolf *et al.* (2008). It works by tracking a set of infrared markers stuck to a surface with at least two infrared cameras and sophisticated proprietary motion tracking software. It does this at a rate of 300 Hz. The Vicon only measures the position vector. Therefore, to get a pose vector including orientation data, trigonometric relationships between the position vectors were used to determine the angular orientation. Given its well-documented measurement accuracy, the measurement results from the Vicon were taken as ground-truth.

The Vicon system used for the test is located in the 3D Human Motion Laboratory on Stellenbosch University's Tygerberg medical campus. It consists of eight infrared cameras arranged around a square on the floor in a configuration

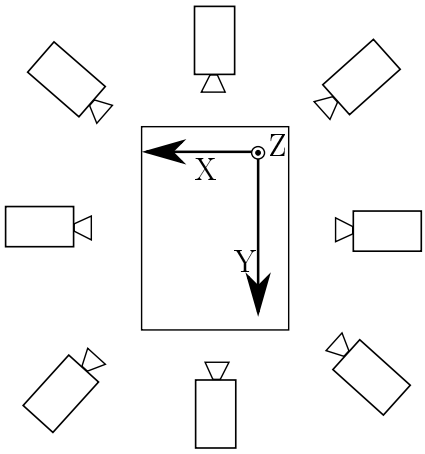


Figure 3.1: Top view of the Vicon motion capture system (not drawn to scale).

that maximises the number of infrared markers visible to each camera at any given point in time. Figure 3.1 shows a diagram of the Vicon system layout.

Before the measurement test commenced, the Vicon system was calibrated using a procedure similar to the CVS's, but the calibration object used is a special 'wand' whose dimensions are pre-programmed into the Vicon system. Infrared markers were then placed on both the calibration board and the CVS's camera to provide pose data on both. Since the Vicon and CVS camera each have their own coordinate systems, having the position and orientation of the CVS's camera available will allow the Vicon's measurements to be related back to the CVS's camera coordinate system during the data processing phase. The markers were placed in such a way that they produce axes that more or less coincide with the Vicon's axis system, thereby simplifying the data processing phase slightly. Figure 3.2 shows the axes for both the CVS and Vicon systems.

Only three markers need to be visible to the Vicon system's cameras for it to record an object's position vector. However, a fourth asymmetrical auxiliary marker was also placed to provide fail-safe orientation data during the data processing phase. Figure 3.3 shows the Vicon marker placements for the camera and the calibration board. These markers were carefully placed by hand in line with one another, but some placement error is inevitable. This placement error offset is taken into account during the data processing phase.

One aspect of the Vicon system to note is that the infrared markers have some high-frequency noise associated with them, which becomes apparent when inspecting the raw data. This noise is inherent to the marker and can be safely filtered out with a zero-lag, second order Butterworth filter. However, to avoid any unknown filtering factors from affecting the data, the raw, unfiltered data is used throughout this project.

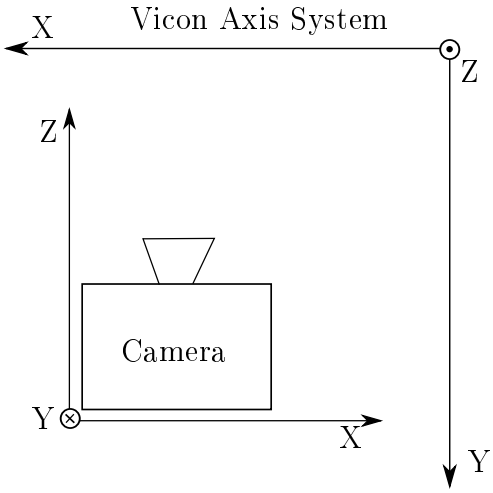
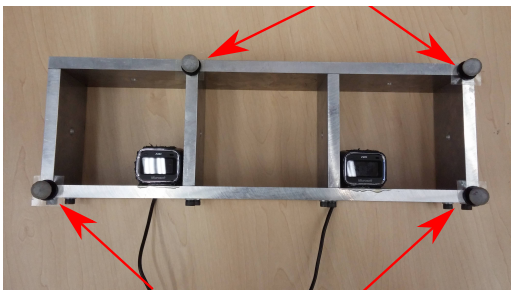
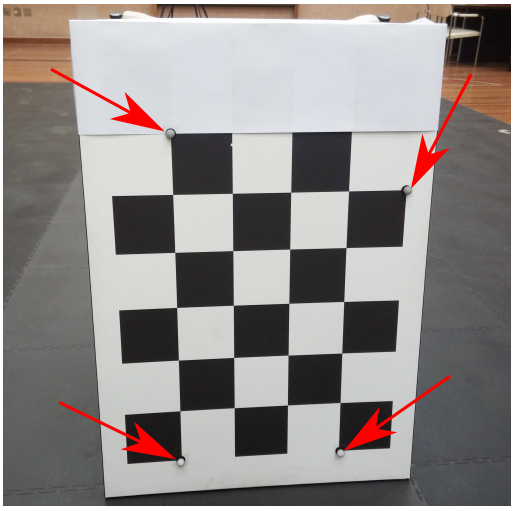


Figure 3.2: The axis orientations of the Vicon and CV systems.



(a) Infrared marker placement on the camera frame. Note that only one of the cameras are used in the CVS.



(b) Infrared marker placement on the chessboard.

Figure 3.3: Diagram of the Vicon marker placement on the calibration board and camera frame.

Computer Vision System Setup

The CVS configuration used during the Vicon test is the exact same as the one described in Section 3.2.

Before the test the CVS's camera was calibrated to determine its intrinsic parameters, including its focal lengths in the x and y directions. Calibration was done with the same board and camera that was used during the Vicon test, against a white, well-lit background. The board was moved to different positions and orientations within view of the camera and roughly 15 still images at a resolution of 640×480 pixels were taken. All of the test video data was captured at this resolution. The camera was then calibrated with this set of images and OpenCV's camera calibration module to produce a camera matrix that gives a reprojection error of approximately 0.21 pixel units.

This project is an initial study into using low-cost cameras to make pose measurements and involved much video processing. It was found that HD video data (720×1280 for the CVS's webcam) produced marginally more accurate pose measurements. However, it was opted to capture video data at a lower resolution of 640×480 to increase the speed at which pose data is extracted from the video data. For example, a single run of the optimisation procedure (discussed in more detail later) using HD video data takes approximately 6.5 hours, while the SD run takes less than 2 hours to complete. Since these data extraction scripts were run multiple times during the CVS's design and testing, using SD video data led to large time savings.

During the test, the CVS camera was placed in a stable aluminium frame and left untouched throughout the test. The laptop was set to capture video at 640×480 pixels, while zoom and autofocus was disabled to keep the camera's focal length constant. Data extraction and pose estimation took place off-line.

3.3.2 Test Procedure

Figure 3.4 shows a picture of the complete test setup. The camera was placed on a chair and the calibration board was held facing the camera. Both were covered with four infrared markers with the Vicon's eight infrared cameras surrounding both the board and camera.

Data capture for both systems was initialised when the Vicon system started recording data. At the same time the calibration board was tilted forward, which allowed the data sets from both systems to be synchronised to a common timeframe during the data processing phase.

During the test it was desirable to record a variety of pose vector combinations to ensure a diverse cloud of data in both measurement sets. This was done by moving the calibration board around the CVS camera's full field of vision and depth of view by hand, while varying the board's orientation.

Each video is approximately 90 seconds long, equating to approximately 2700 measurement samples per test, though it was decided that only 2400 of

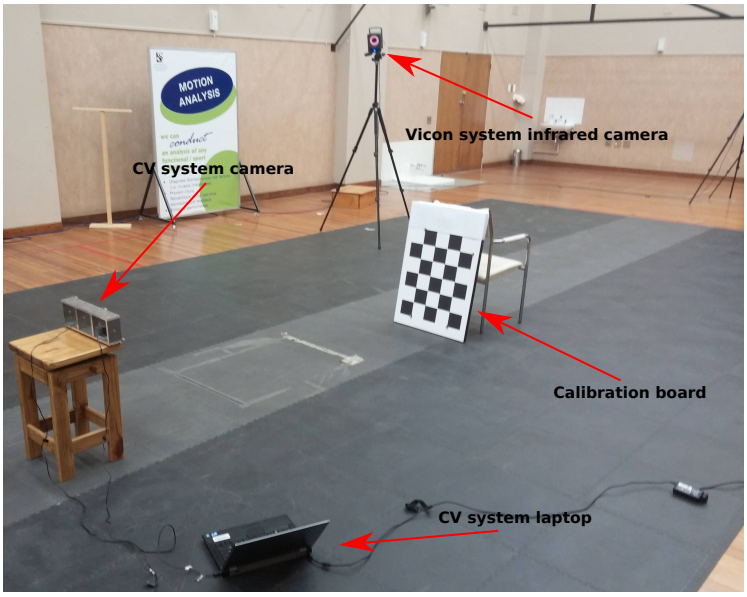


Figure 3.4: Picture of the Vicon test layout.

these points will be used since approximately 300 samples gave invalid readings. An invalid reading is given when the CVS cannot detect all of the expected corner features on the calibration board and as a result cannot estimate the board's pose for that image sample.

The measurement test produced two sets of data: one measurement set from the Vicon system, which can be used as ground-truth reference measurements, and another set recorded by the CVS. These data sets allows the CVS's measurement error to be determined by comparing the two sets of measurement data. These sets were also used to provide training and validation data sets for the error prediction model discussed in Chapter 4.

3.3.3 Data Processing

During the measurement test the CVS only recorded video data, leaving the feature data extraction and pose estimation to be performed off-line during the data processing phase.

The Vicon system's data required little processing, since most of the data was generated in real-time and was optimised by the Vicon software. However, some work was done to fix invalid measurements that were introduced when not enough Vicon cameras had a clear view of the infrared markers for a few consecutive frames. These measurements would normally be discarded, but since the measurements are time dependant, they were corrected by means of interpolation instead.

Processing the CVS data involved several steps, the first of which was to make the data sets directly relatable by centering both data sets around a common axis system. Furthermore, the marker placement, measurement bias

and other error offsets, e.g. the placement and orientation offset error between the Vicon markers and chessboard and camera frame surfaces, were determined while simultaneously optimising the camera matrix's focal lengths to produce the most accurate CVS pose measurement results. Each of these aspects are discussed here in more detail.

Reducing Vicon Sample Speed

The Vicon system captures data at 300 Hz and the CVS at 30 Hz. Therefore, the Vicon's framerate had to be downsampled to match the CVS's framerate so that the two data sets could directly be related.

Downsampling by a factor of 10 could have been accomplished by taking 1 out of every 10 samples and discarding the rest. However, to avoid discarding data that may have contained valuable information and trends, an interpolation approach was opted for instead. The process involved averaging the Vicon data set in intervals of 10 samples, thereby downsampling the Vicon by a factor of 10 and matching the CVS's framerate.

During the test, the calibration board was tilted forward to signal the start of a measurement. The two data sets were synchronised with respect to time by matching these initial peaks in the measurement data.

Rotating and Centering the Camera and Chessboard Data

During the measurement test, four infrared markers were placed on the CVS camera's frame to provide data on its placement within the Vicon's axis system. The markers were roughly placed along the frame's x and y axes and coincided with the chessboard's axis system. Since the CVS's camera remained still during the test, it is most convenient to centre the CVS's camera and pose measurement data around the Vicon's axis system.

With the CVS's camera placement and orientation in the Vicon coordinate system known, relocating and reorientating the CVS's camera pose data became a relatively simple task. Since the CVS's position and orientation remained constant throughout the measurement test, its data could be centred around the Vicon axis system by simply subtracting the CVS's placement coordinates from each of the measurement vectors produced by the CVS.

With the CVS's camera axis system now centred around the Vicon system's origin, the pose data for the chessboard acquired by the CVS and the Vicon system are now directly relatable to one another.

Camera Matrix Parameter Optimisation

Before the measurement test took place, the CVS camera was calibrated using OpenCV's camera calibration toolbox. The calibration procedure provides a good estimate of the intrinsic parameters of the camera, which includes the focal length, lens distortion and principle points. However, given the lack of

reference three-dimensional data, it is only an estimate of the intrinsic parameters. Using the ground-truth pose data from the Vicon system allowed the intrinsic parameters to be determined more accurately.

Since the Vicon's infrared markers were not placed exactly on the calibration board's x and y axes, the infrared marker placement offset error were also taken into account and determined. This offset error accounts for the marker placement error, as well as any other constant measurement bias that may have been introduced into the CVS's measurements.

This presents a circular optimisation problem: the focal length will affect the perceived error offset, while the error offset will affect the CVS pose estimates, which in turn affects the ideal focal length. To find the offset and optimum focal length, a dual optimisation strategy was implemented.

First, the optimisation algorithm is formulated. Suppose $\boldsymbol{P}^*$ denotes the six-dimensional pose vector of the calibration board as produced by the Vicon system, while $\bar{\boldsymbol{P}}$ is the constant error offset vector. The subscripts c and b represent the data sets of the camera and calibration board respectively. $\boldsymbol{\epsilon}$ refers to the error vector between the Vicon and CVS's measurements that needs to be determined and $\boldsymbol{P}(f_x, f_y)$ is the pose vector measured by the CVS's camera as a function of its focal lengths, f_x and f_y. The Vicon pose and offset vectors are given by Equations 3.5 and 3.6. Figure 3.5a shows the relation for the $\boldsymbol{P}^*$ vector and Figure 3.5b shows the marker placement error offset for the $\bar{\boldsymbol{P}}$ vector for a corner on the calibration board.

$$\boldsymbol{P}^* = \boldsymbol{P}^*_{\mathrm{b}} - \boldsymbol{P}^*_{\mathrm{c}} \tag{3.5}$$

$$\bar{\boldsymbol{P}} = \bar{\boldsymbol{P}}_{\mathrm{b}} - \bar{\boldsymbol{P}}_{\mathrm{c}} \tag{3.6}$$

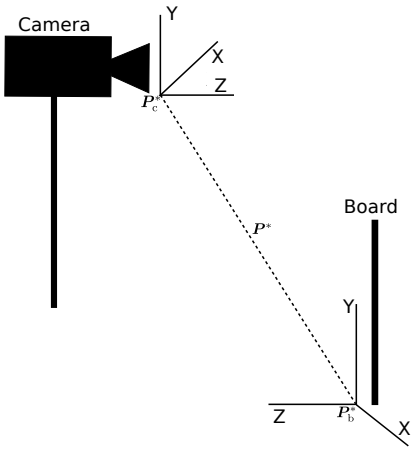
The equation for the pose estimate from the CVS is given by

$$\boldsymbol{P}(f_x, f_y) = \boldsymbol{P}^* - \bar{\boldsymbol{P}} + \boldsymbol{\epsilon}. \tag{3.7}$$

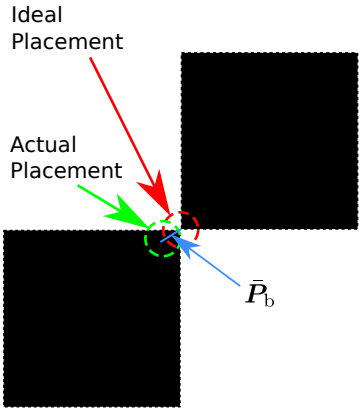
Equation 3.7 indicates that the CVS's pose measurements, $\boldsymbol{P}(f_x, f_y)$, are given by the Vicon's measurement $(\boldsymbol{P}^* - \bar{\boldsymbol{P}})$ plus some error vector, $\boldsymbol{\epsilon}$, which represents the CVS's measurement error. Equation 3.7 can be simplified to the form given in Equation 3.8, which forms the basis of the optimisation algorithm moving forward.

$$\boldsymbol{P}(f_x, f_y) = (\boldsymbol{P}^*_{\mathrm{b}} - \bar{\boldsymbol{P}}_{\mathrm{b}}) - (\boldsymbol{P}^*_{\mathrm{c}} - \bar{\boldsymbol{P}}_{\mathrm{c}}) + \boldsymbol{\epsilon} \tag{3.8}$$

The next step is then to determine the constant perceived offset for a given focal length combination. The focal length is initialised with the focal lengths found during calibration (approximately 700 pixel units). By summing the samples, the offset, $\bar{\boldsymbol{P}}$, can be determined with



(a) Diagram showing the relation between the pose vector between the camera and board within the Vicon axis system.



(b) Diagram showing the marker placement error offset in the calibration board.

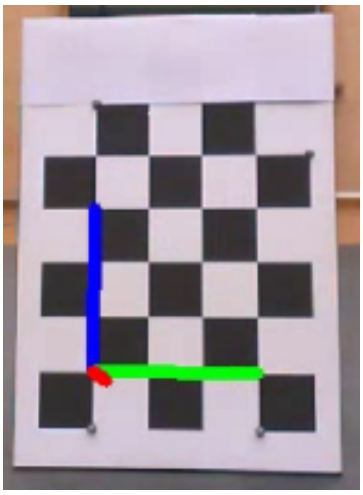
Figure 3.5: Diagrams of the relationships between the pose vectors given by the Vicon and CVS.

$$M\bar{\boldsymbol{P}} = \sum_{i=1}^{M} \boldsymbol{P}_i(f_x, f_y) - \sum_{i=1}^{M} (\boldsymbol{P}^*_{\mathrm{b},i} - \boldsymbol{P}^*_{\mathrm{c},i}) + \mathbb{E}\left[\sum_{i=1}^{M} \boldsymbol{\epsilon}_i\right]. \tag{3.9}$$

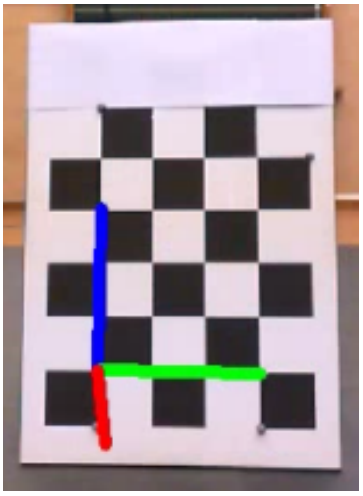
In Equation 3.9, i is the pose vector sample number and M represents the number of samples in the data set. At this point, it is assumed that the error vector, $\boldsymbol{\epsilon}$, has an expected mean value of zero, eliminating its sum and leading to Equation 3.10. This assumption was made in the hope that the CVS will on average produce a small error and it is verified in Section 3.4.2.

$$\bar{\boldsymbol{P}} = \frac{1}{M}\left(\sum_{i=1}^{M} \boldsymbol{P}_i(f_x, f_y) - \sum_{i=1}^{M} (\boldsymbol{P}^*_{\mathrm{b},i} - \boldsymbol{P}^*_{\mathrm{c},i})\right) \tag{3.10}$$

Using the constant $\bar{\boldsymbol{P}}$ produced by Equation 3.10, it is now possible to minimise the error vector, $\boldsymbol{\epsilon}$, of Equation 3.8 by varying the focal lengths, f_x and f_y. The optimum focal lengths were found by setting up a $3 \times M$ cell error matrix, consisting of the three translation dimensions, $[x\ y\ z]^T$, and find the optimal focal length combination by minimising the matrix's two-norm. The optimisation is based only on the three position dimensions since it was found that the orientation dimensions contain a large amount of noise and including them in the optimisation added too much variability to the procedure, leading to instability. Figure 3.6 demonstrates this by comparing two images taken during the Vicon test: one using the optimum focal length calculated using only the position dimensions and the other using all six dimensions.



(a) Axis system drawn with focal length calculated from using the position dimensions.



(b) Axis system drawn with focal length calculated from using all six dimensions.

Figure 3.6: Images demonstrating the optimisation procedure's sensitivity to the orientation dimensions.

In Figure 3.6, the CVS drew axes onto the calibration board by using the pose it calculated. It can be seen that the axes differ from one another, indicating that the optimisation is sensitive when using the orientation dimensions in the optimisation procedure.

The error matrix is produced by

$$\boldsymbol{\epsilon} = \boldsymbol{P}(f_x, f_y) - \boldsymbol{P}^* - \bar{\boldsymbol{P}}. \tag{3.11}$$

Using the $3 \times n$ error matrix from Equation 3.11, an error vector is generated by summing the error matrix along its columns. Only the position dimensions were relevant to the optimisation procedure, so the error matrix only contains 3 dimensions. The optimum focal lengths were then found by taking the focal length combination that produces the smallest two-norm of the sample error vector $\boldsymbol{\epsilon}$. The minimisation equation is given by

$$\hat{f}_x, \hat{f}_y = \min_{\hat{f}_x, \hat{f}_y} \sum_i \|\boldsymbol{\epsilon}_i\| , \tag{3.12}$$

where $\hat{f}_x$ and $\hat{f}_y$ refer to the optimum focal lengths. With the optimal focal lengths now determined, the procedure is repeated again from Equation 3.8, finding a new offset for the new focal length combination. The optimisation procedure is iterated a number of times, minimising the total error norm. After the optimum focal length combination was found, that combination was used in the camera matrix that extracted the pose data from the video data.

The optimisation procedure is summarised in Algorithm 1. Here, the index i refers to a frame from the video data.

Algorithm 1 Optimise Camera Focal Length Parameter

Initialise variables:
$\hat{f}_x \leftarrow 700.0$
$\hat{f}_y \leftarrow 700.0$
$i \leftarrow 0$
while $\|\boldsymbol{\epsilon}\| > \|\boldsymbol{\epsilon}_{min}\|$ **do**
  Find $\bar{\boldsymbol{P}}$ for f_x^i and f_y^i (Equation 3.10)
  Determine $\|\hat{\boldsymbol{\epsilon}}\|$ for the new $\bar{\boldsymbol{P}}$ (Equation 3.11)
  if $\|\hat{\boldsymbol{\epsilon}}\| < \|\boldsymbol{\epsilon}\|$ **then**
    $\boldsymbol{\epsilon} \leftarrow \hat{\boldsymbol{\epsilon}}$
    $\hat{f}_x \leftarrow f_x^i$
    $\hat{f}_y \leftarrow f_y^i$
  end if
  $i \leftarrow i + 1$
end while
$\hat{\boldsymbol{P}} \leftarrow \boldsymbol{P}(\hat{f}_x, \hat{f}_y)$

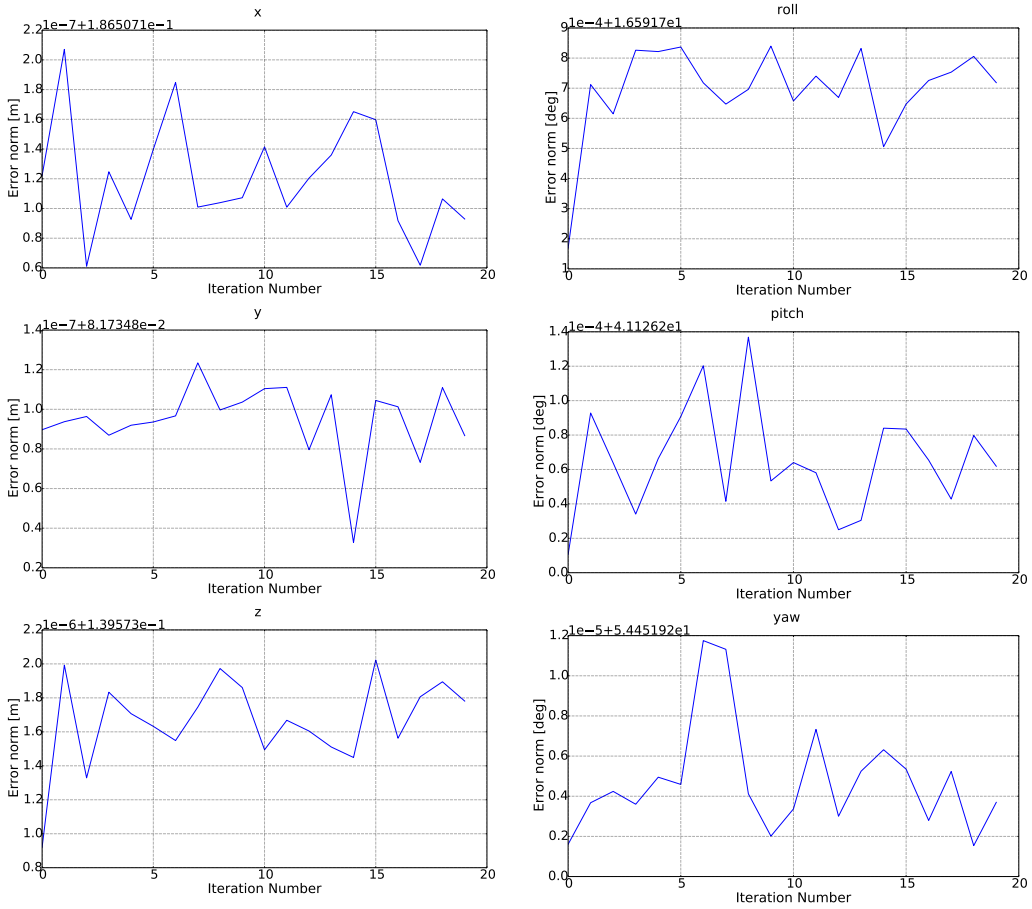
3.4 Results

Several sets of data were produced during the data processing phase. The results are presented and discussed in this section. They include evidence of error convergence for the optimisation procedure, as well as proof that the error $\boldsymbol{\epsilon}$ is indeed distributed around zero and also roughly normally distributed. Lastly, the new focal length combination is given and discussed, followed by the pose measurement accuracy results of the CVS.

3.4.1 Evidence of Convergence

To show that the optimisation procedure worked as expected, minimising the error in each dimension, the minimum error norm after each iteration of the process is plotted in Figure 3.7 and shows the error over 20 iterations of the optimisation procedure. It was found in Figure 3.8 that the process converges to a constant value after approximately 12 iterations. Figure 3.8 plots the total norm reduction per iteration.

The graphs in Figure 3.7 show that the error is reduced in the x dimension, remains roughly the same in the y and yaw dimensions and is worsened in the rest. It was expected that the position dimensions will be improved, since the optimisation weight was placed on reducing their error norm. It is unfortunate, however, that the y dimension's norm did not see improvement. The fact that the z dimension got slightly worse does not necessarily condemn the optimisation procedure, since it can be seen in Figure 3.8 that the total position norm does get reduced. Conversely, the orientation dimensions were not expected to see significant improvement since no weight was placed on



(a) Error convergence plots for the position dimensions.

(b) Error convergence plots for the orientation dimensions.

Figure 3.7: Plots showing the error in each dimension during for each iteration of the error minimisation procedure.

them during optimisation.

The plots in Figures 3.7 exhibit some spiky behaviour with large variations, sometimes growing larger than its initial value. This is not an issue, however, since Figure 3.8 shows that the total error norm decreases as the iterations increase, which is what was expected. It can also be seen that the total norm regularly stays constant for more than one iteration. It must be kept in mind at this point that it is not only the focal length that is being optimised, but the offset vector as well. Therefore, while the offset vector is being refined between iterations, it may affect the minimum norm values.

Overall it can be concluded that Figures 3.7 and 3.8 exhibits sufficient evidence of convergence for the CVS's pose measurement error.

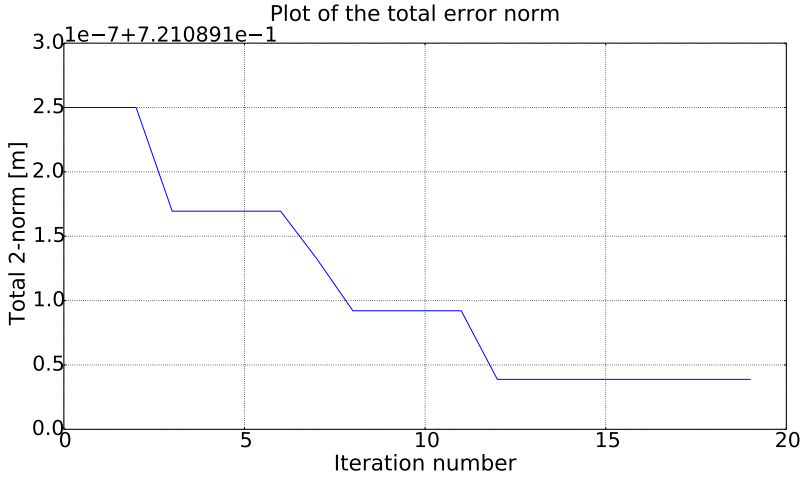


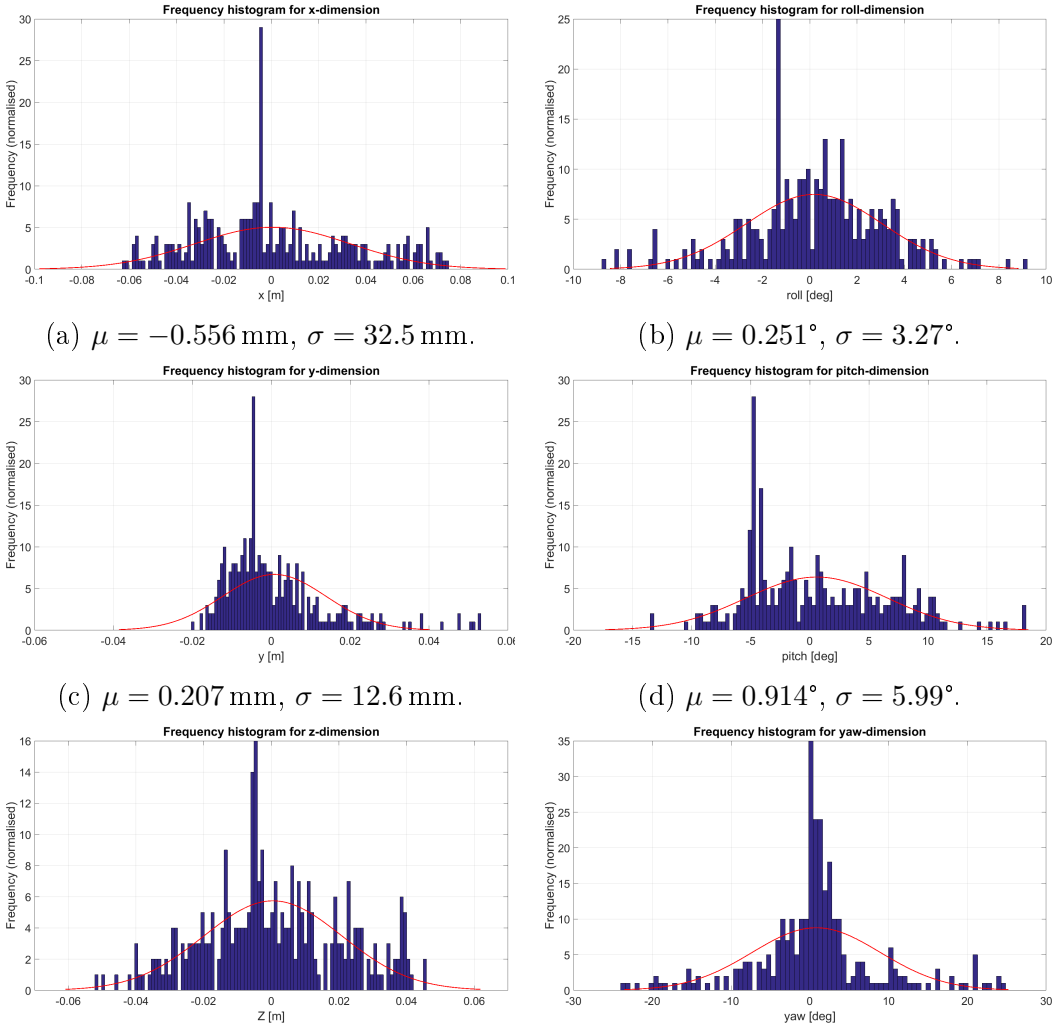
Figure 3.8: Plot of the minimum norm values for the position dimensions for the 20 iterations of the optimisation procedure.

3.4.2 Test for Zero Average Error

To verify whether the error in $\boldsymbol{\epsilon}$ is indeed distributed around a zero mean value as assumed in Equation 3.10, a χ^2 (chi-squared) test could have been used. However, it was found that the sample size of the data set was too large, where the large number of samples can mislead the χ^2 test to believe the data contains skewness and it is known that skewness in the data can have a large impact on the χ^2 probability estimate (Wackerly *et al.*, 2007). Therefore, a graphical approach was taken. This was done by drawing frequency histograms of the error, $\boldsymbol{\epsilon}$, in all six dimensions, along with a normal distributions drawn using each dimension's mean and standard deviation. This is done to check if the error data is normally distributed, since this will allow additional tools to be used and justify future assumptions. The plots are given in Figure 3.9.

From Figure 3.9 it can be seen that all the plots are roughly centred around zero and are very nearly normally distributed. The x dimension displays the largest standard deviation of 32.2 mm followed by the z dimension. This implies that the x dimension's estimates are the least accurate. The standard deviations of the other dimensions are within the range that was expected of the system.

In all, the frequency histogram and normal distribution plots of the errors in Figure 3.9 show that the assumption made in Equation 3.10 (that the errors are distributed around zero) is indeed a valid one. Furthermore, the error data looks to be roughly normally distributed allowing other statistical tools to be used on the data set.



(a) $\mu = -0.556\,\text{mm}$, $\sigma = 32.5\,\text{mm}$.

(b) $\mu = 0.251°$, $\sigma = 3.27°$.

(c) $\mu = 0.207\,\text{mm}$, $\sigma = 12.6\,\text{mm}$.

(d) $\mu = 0.914°$, $\sigma = 5.99°$.

(e) $\mu = -0.253\,\text{mm}$, $\sigma = 20.8\,\text{mm}$.

(f) $\mu = 0.320°$, $\sigma = 8.11°$.

Figure 3.9: Frequency histograms of the error data in each dimension. Each dimension's mean value (μ) and standard deviation (σ) is given in the captions.

3.4.3 Optimum Focal Lengths and Offset

As a result of the focal length and offset optimisation procedure, the optimal focal lengths, $\hat{f}_x$ and $\hat{f}_y$, were found to be 694 and 704 respectively after 20 iterations of the algorithm. This is approximately the 700 which was determined by the calibration toolbox. Note that these units are given in camera pixel units and not millimetres. The optimal offset, $\hat{\hat{\boldsymbol{P}}}$, was found to be

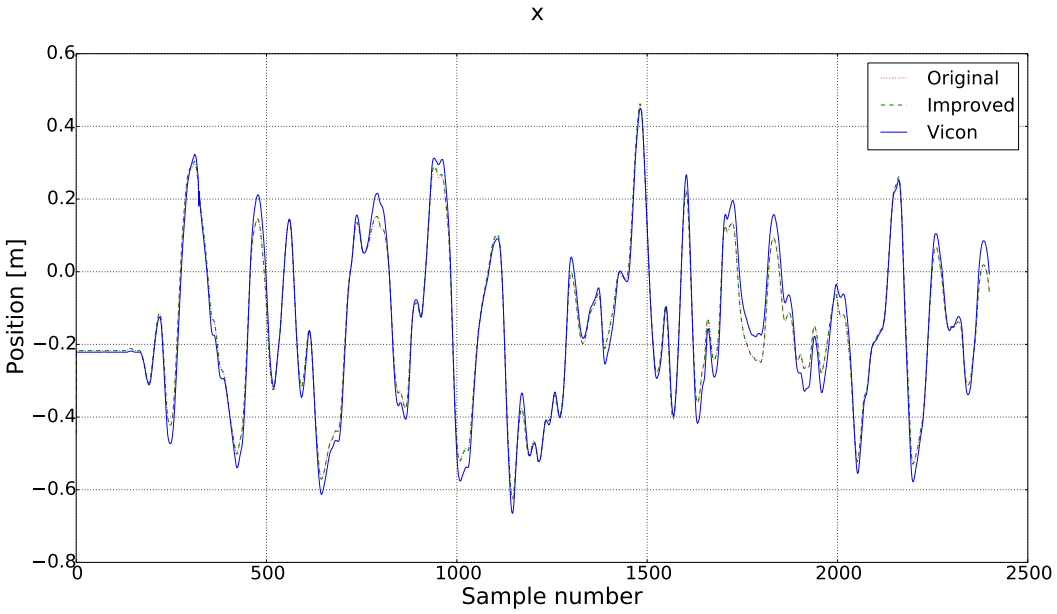
$$\hat{\hat{\boldsymbol{P}}} = \begin{bmatrix} 283.8\,\text{mm} & 56.16\,\text{mm} & -57.44\,\text{mm} & 179.8^\circ & -1.305^\circ & -178.8^\circ \end{bmatrix}^T. \tag{3.13}$$

The values in $\hat{\hat{\boldsymbol{P}}}$ contain any constant error bias as well as marker placement errors that may have been introduced to the Vicon test during the measurement process. The large offsets of $\pm 180^\circ$ in the *roll* and *yaw* dimensions can be explained by the differing axis orientations of the CVS camera and Vicon systems where the axes needed to be rotated to coincide with one another. This indicates that the offset was correctly calculated and was working as expected. The relatively large offset in the x dimension was initially surprising, since this indicated that there was a large constant level of error bias introduced to the x dimension's measurements. However, this would make sense if the CVS's camera was placed approximately 280 mm from the Vicon's axis centre. It turns out this was indeed the case.

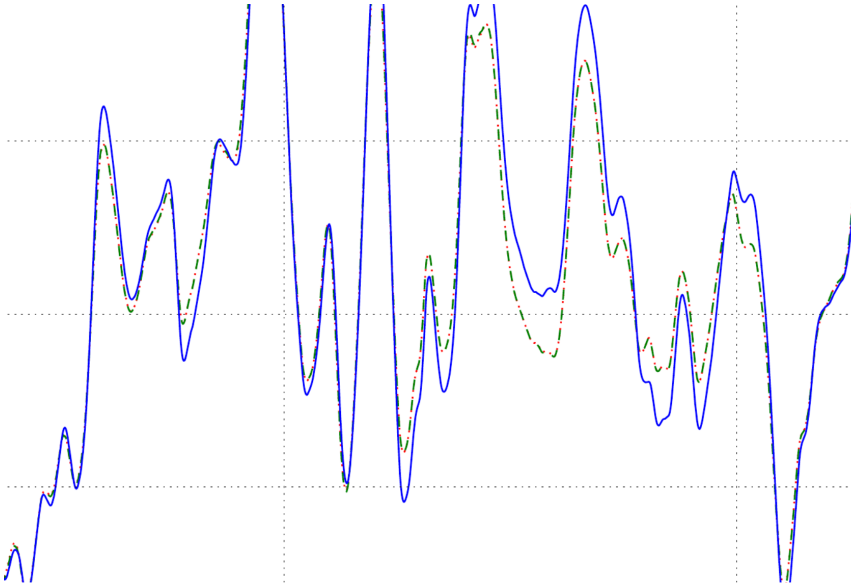
In all, the optimisation procedure reduced the error norm by approximately 0.1 % showing an overall, albeit minimal, reduction in the error vector magnitude when compared to the measurements made with the original focal length combination. This indicates that OpenCV does a good job of estimating the camera's intrinsic parameters without any reference measurements.

Figures 3.10 to 3.15 show the results of the Vicon measurements compared to the original and improved CVS measurements in all six dimensions. In the plots in Figures 3.10 to 3.15, it is difficult to see any significant difference between the improved and original plots. To make these differences more obvious to the reader, each plot is accompanied by a zoomed-in figure of each dimension's plot.

It can be seen that there is some improvement in all of the dimensions. However, in some cases the improvement in one section of the data set is negated by worse estimates in another. This can be attributed to the optimisation process, where an improvement at timeframe t_i in the x dimension, for example, may lead to a worse estimate at time t_i in the *roll* dimension. However, as the reduction in the two-norm magnitude of the error proves, there is an overall reduction in the error with the optimised data set. It can therefore be concluded that the optimisation procedure did indeed function as expected, producing a focal length combination which led to a more accurate pose estimate from the CVS.

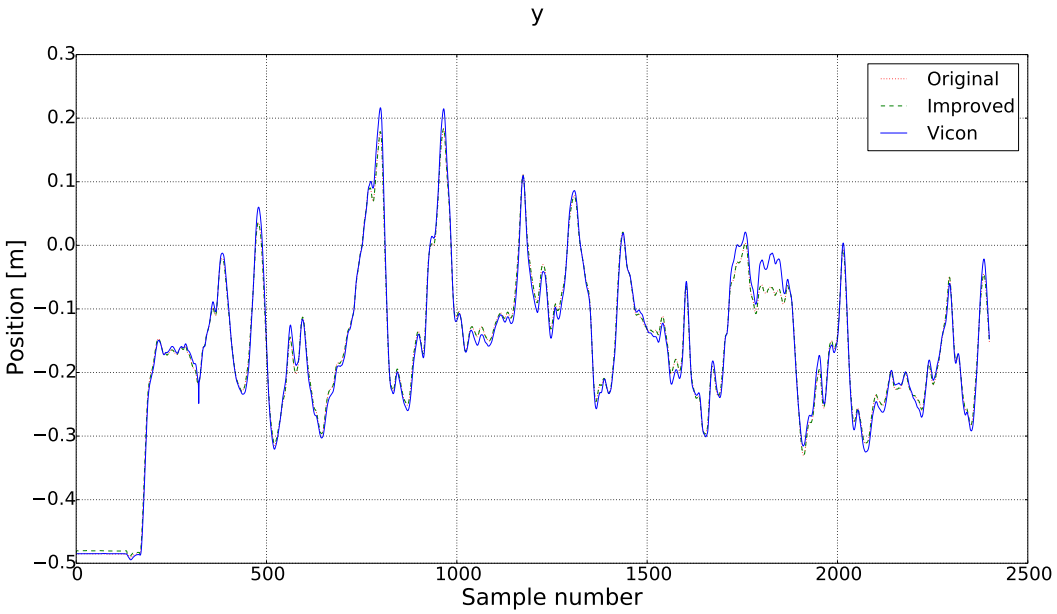


(a)



(b)

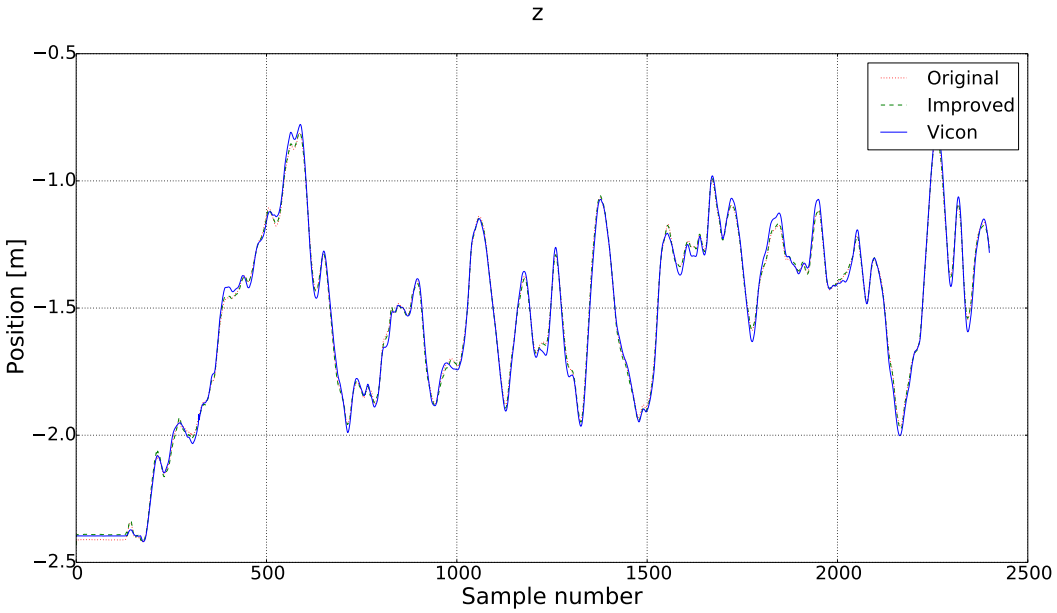
Figure 3.10: Plot comparing the Vicon, original and improved CVS measurements in the x dimension, as well as a zoomed-in figure highlighting the detail.



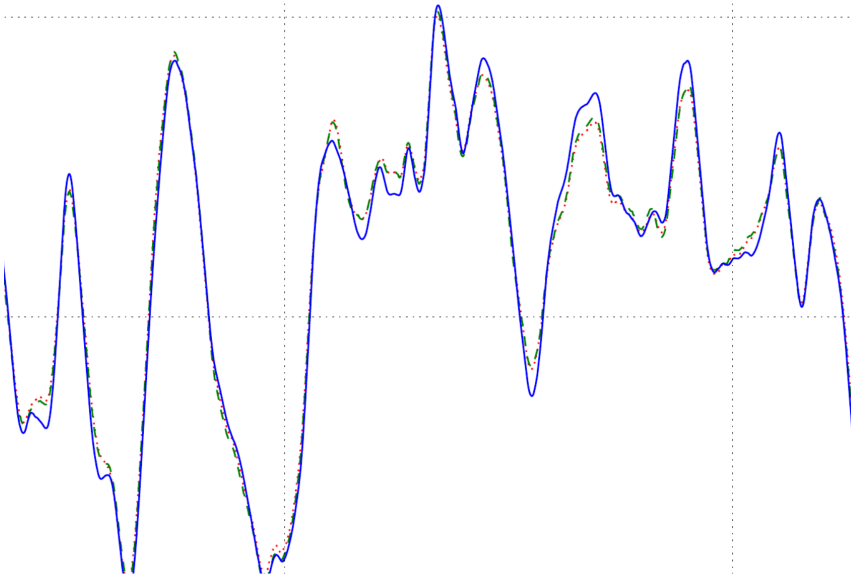
(a)

(b)

Figure 3.11: Plot comparing the Vicon, original and improved CVS measurements in the y dimension, as well as a zoomed-in figure highlighting the detail.

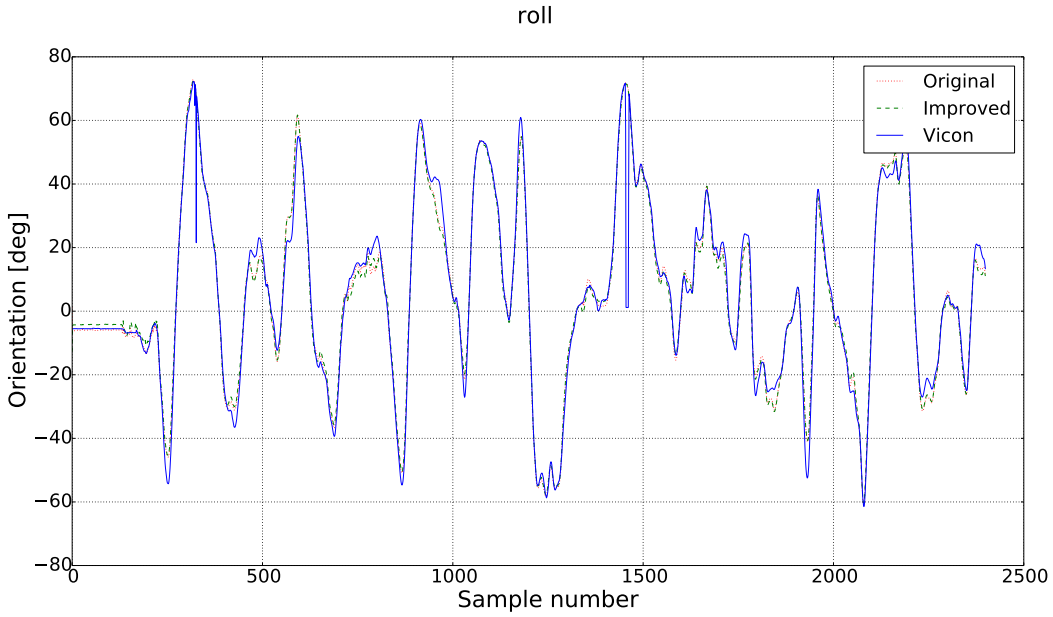


(a)

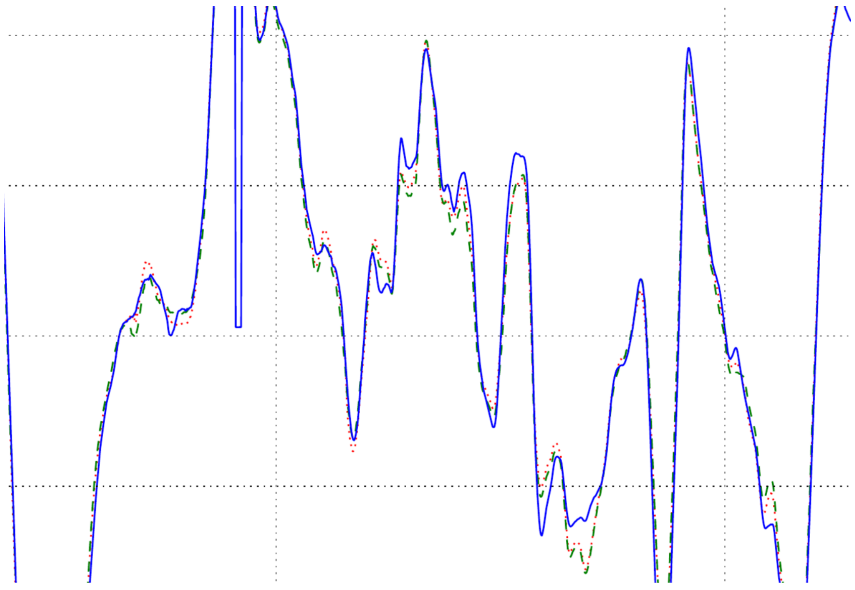


(b)

Figure 3.12: Plot comparing the Vicon, original and improved CVS measurements in the z dimension, as well as a zoomed-in figure highlighting the detail.

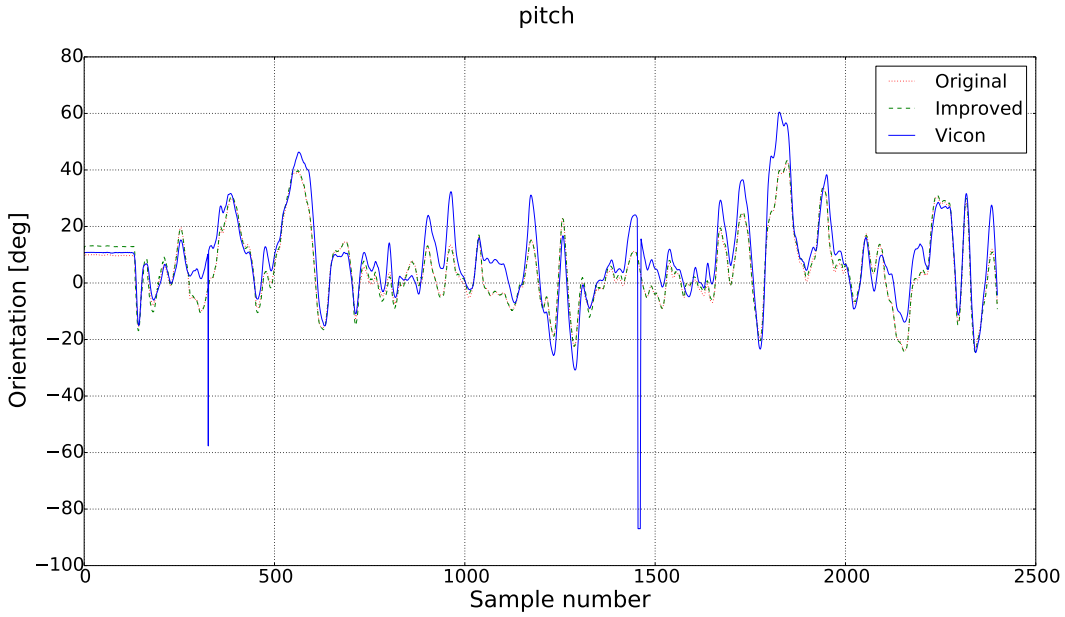


(a)

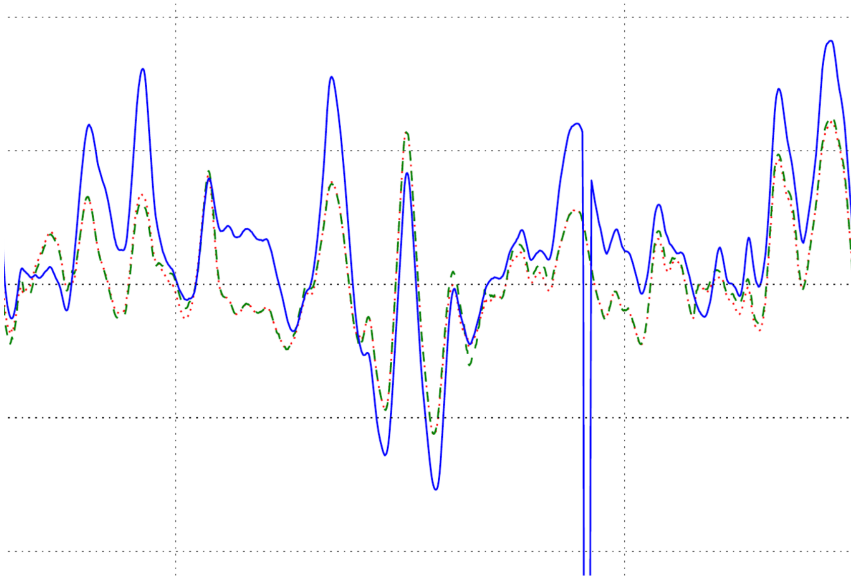


(b)

Figure 3.13: Plot comparing the Vicon, original and improved CVS measurements in the *roll* dimension, as well as a zoomed-in figure highlighting the detail.

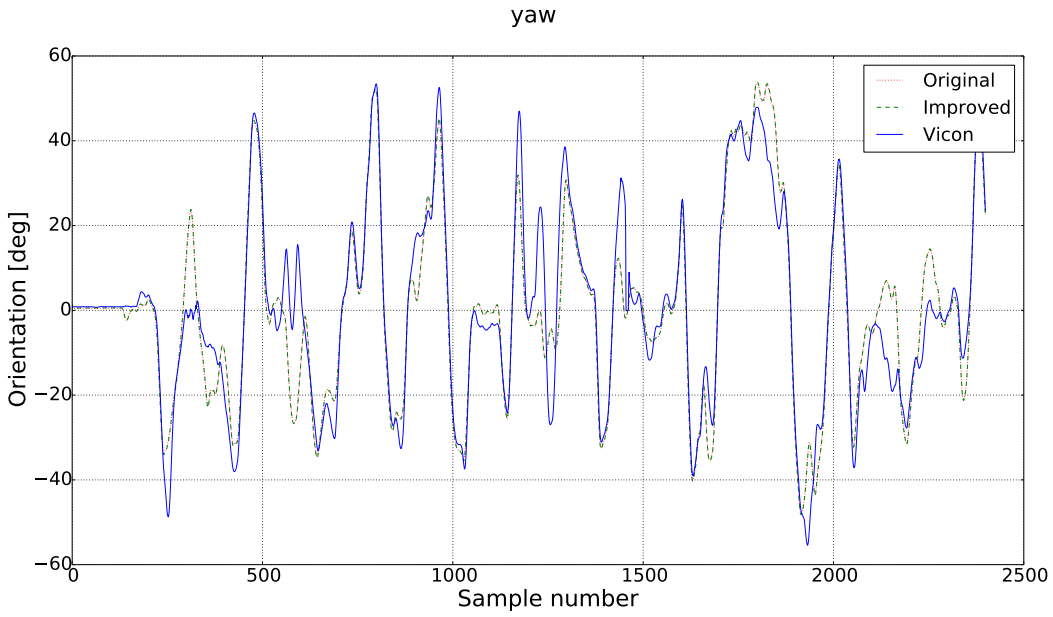


(a)

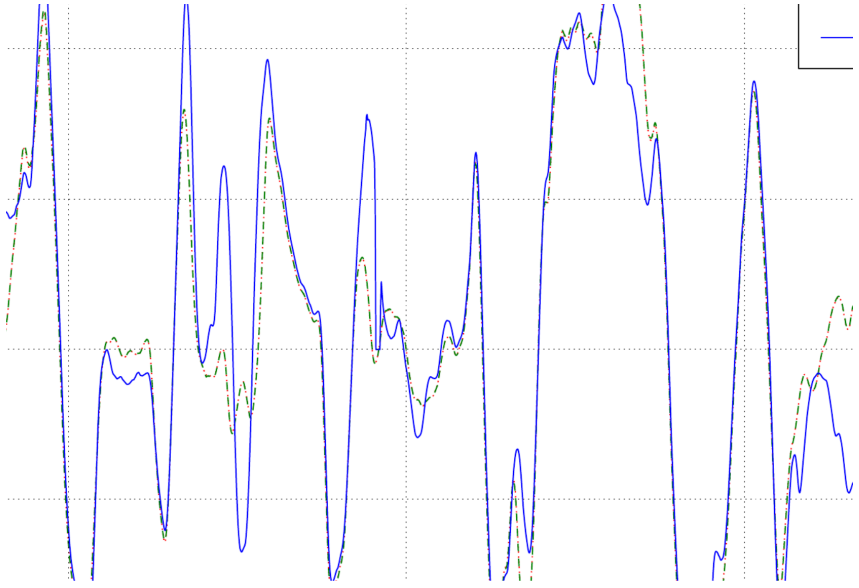


(b)

Figure 3.14: Plot comparing the Vicon, original and improved CVS measurements in the *pitch* dimension, as well as a zoomed-in figure highlighting the detail.



(a)



(b)

Figure 3.15: Plot comparing the Vicon, original and improved CVS measurements in the *yaw* dimension, as well as a zoomed-in figure highlighting the detail.

3.4.4 Computer Vision System Accuracy

Determining the accuracy of a multi-dimensional data model is often a complex task, but since it was found that the error $\boldsymbol{\epsilon}$ is normally distributed about zero, it is possible to use the covariance matrix to check the interdimensional variance and dependence. If the off-diagonal elements of the covariance matrix are sufficiently small relative to the diagonal elements, it can be assumed that the dimensions are strong enough independent of one another. The covariance matrix, Σ, is given by Equation 3.14.

$$\Sigma = \begin{bmatrix} \mathbf{1.05 \times 10^{-3}} & 2.10 \times 10^{-4} & 8.91 \times 10^{-4} & 4.64 \times 10^{-2} & 4.14 \times 10^{-2} & 2.06 \times 10^{-2} \\ 2.10 \times 10^{-4} & \mathbf{1.60 \times 10^{-4}} & 7.31 \times 10^{-5} & 9.11 \times 10^{-3} & 3.30 \times 10^{-2} & -1.66 \times 10^{-2} \\ 8.91 \times 10^{-4} & 7.31 \times 10^{-5} & \mathbf{4.32 \times 10^{-4}} & -2.39 \times 10^{-3} & 3.34 \times 10^{-2} & -8.08 \times 10^{-3} \\ 4.64 \times 10^{-2} & 9.11 \times 10^{-3} & -2.39 \times 10^{-3} & \mathbf{-1.07 \times 10^{1}} & 7.22 \times 10^{0} & 1.26 \times 10^{0} \\ 4.14 \times 10^{-2} & 3.30 \times 10^{-2} & 3.34 \times 10^{-2} & 7.22 \times 10^{0} & \mathbf{3.59 \times 10^{1}} & 2.91 \times 10^{0} \\ 2.06 \times 10^{-2} & -1.66 \times 10^{-2} & -8.08 \times 10^{-3} & 1.26 \times 10^{0} & 2.91 \times 10^{0} & \mathbf{6.59 \times 10^{1}} \end{bmatrix} \tag{3.14}$$

The matrix Σ displays large off-diagonal elements, indicating that there are strong interdimensionally dependant relationships. This dependence is further demonstrated by the plots in Figure 3.16. Here 300 random data points and their measurement errors are sorted from large to small and the errors from the six dimensions are plotted over the z dimension's measurements.

One would expect the error to gradually increase as the distance between the camera and calibration board increases. However, it can be seen from Figure 3.16, with high error points seemingly randomly intersperesed between low error points, that this is not the case. This indicates that the error is dependant on all six dimensions and demonstrates the error data's complexity.

These plots, as well as the covariance matrix Σ, show that there is no clear indication on the CVS's accuracy, since it is a function of the pose of the calibration board relative to the CVS's camera.

3.5 Conclusion

In this chapter, the design and layout of a computer vision measurement system (CVS) was discussed. The system consists of hardware and software components and the details of both were discussed. The system was tested in an indoor measurement facility to determine its measurement accuracy. With the ground-truth measurements produced by this indoor system, it was possible to optimise the intrinsic camera parameters and improve the accuracy of the CVS's pose measurement data.

The CVS's pose measurement error is not constant over all of the poses and it is therefore necessary to model this error as a function of the calibration board's pose relative to the CVS's camera.

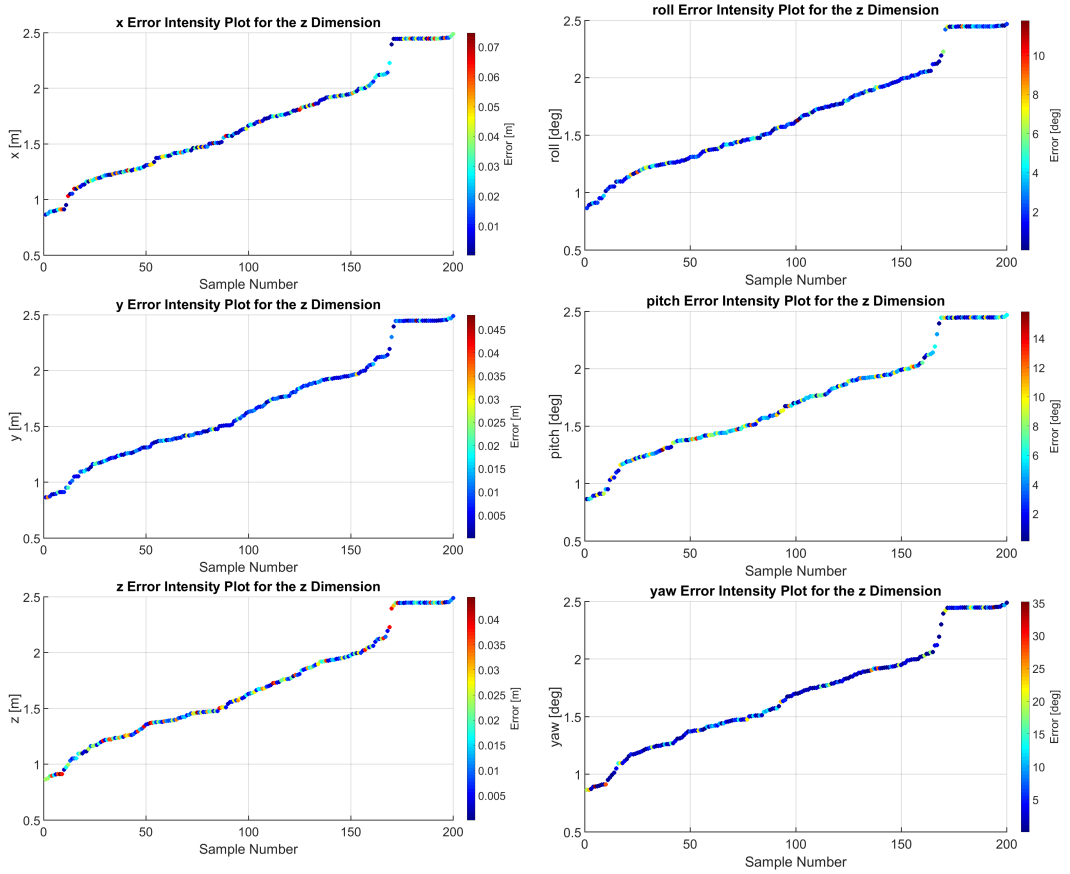


Figure 3.16: Plots demonstrating the complexity of the error data for the z dimension. Note that the error intensity does not change in a predictable, linear manner as the distance from the camera increased.

Chapter 4

Computer Vision System Pose Measurement Error

4.1 Introduction

The main objective of this project is to determine the pose estimation accuracy of a quadcopter's on-board sensor suite. The pose estimation must be performed in the outdoors to provide the quadcopter access to its GPS readings, which is important data for an outdoor quadcopter since it provides absolute position data which can be used to offset the sensor drift introduced by integrating other sensor data. To this end, a computer vision pose measurement system (CVS) which extracts six-dimensional pose data from a calibration object in the system's view, was designed and implemented. However, the CVS's measurement accuracy was first determined before it could be used to make pose measurements,

In Chapter 3 it was found that the pose measurement error of the CVS is characterised by high interdimensional dependence, as demonstrated by the covariance matrix of the CVS's measurement error data in Equation 3.14 and the plots in Figure 3.16. This means that the CVS's pose measurements and corresponding measurement errors are constantly varying according to the relative pose between the calibration board and the CVS's camera. Also, despite using the RANSAC algorithm to filter out outlier data points, the CVS's measurement data is still fairly noisy, particularly in the orientation dimensions.

In the context of this project it is important to know what the accuracy of a pose measurement sample is. This makes it necessary to be able to determine the measurement error of any arbitrary pose measurement sample recorded by the CVS. It was decided that a machine learning method will be employed to accomplish this. Machine learning is where a computer model is trained to recognise patterns within a set of input data and output information on the patterns which are of interest to the model's designer.

This chapter sets out to explain the process behind making a machine

learning model that can estimate with reasonable accuracy what the CVS's pose measurement error for any arbitrary input measurement vector would be. This chapter has been presented at the SolarPACES 2015 conference in Cape Town and a conference proceedings article has been accepted, pending publication (Lock *et al.*, 2015).

The chapter starts with some background information on model selection, design and training. The validation phase of the trained model is then discussed and its accuracy is presented, followed by a brief discussion on the resulting model.

4.2 Model Design

This section discusses the design aspects of the machine learning model used for this project. It begins by giving some background information on the processes involved in designing such a model, including model type selection, training and validation. These aspects are discussed in more detail here.

4.2.1 Background

Designing a machine learning model can be performed in three basic steps. These steps are:

Step 1 Select the machine learning model type.

Step 2 Train the model.

Step 3 Validate the trained model's accuracy.

First, the model type is selected. As stated previously, the CVS's pose measurement data is fairly complex, interdimensionally dependant and high dimensional. A neural network was therefore selected as a machine learning model basis. It has previously been shown by Tu (1996) that neural networks excel at handling complicated data and detecting and extracting complex, non-linear relationships within the input data. Furthermore, the radial basis function neural network (RBFNN) was selected as the network topology. RBFNNs provide superior results when compared to traditional feed-forward networks when noisy data sets are used, as shown by Xie *et al.* (2011). The data is not time dependant, making the recurrent neural network topology unnecessarily complex and costly for this application. An RBFNN is described by the relation given by Equation 2.5 and is repeated again in Equation 4.1.

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{M} \lambda_j \phi(||\boldsymbol{x}_i - \boldsymbol{x}_j||) \tag{4.1}$$

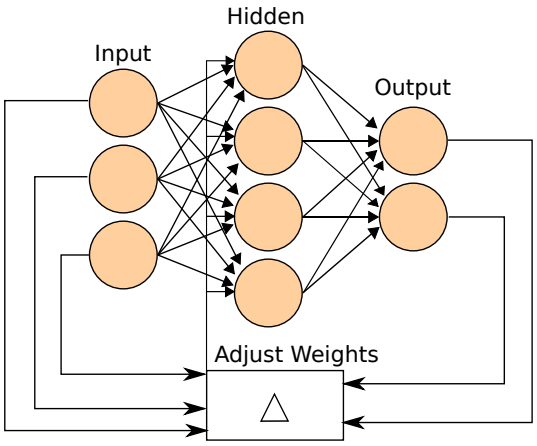


Figure 4.1: A figure of a neural network implementing the backpropagation procedure. Adapted from Wikimedia Commons (2006).

Here, ϕ is a function of the Euclidean distance between the radial basis centres, $\boldsymbol{x}_j$, and the sample points, $\boldsymbol{x}_i$ and is given by

$$\phi(\boldsymbol{r}) = e^{(\frac{r}{\epsilon})^2}. \tag{4.2}$$

Next, model training takes place where the network is trained with a supervised learning scheme, such as the backpropagation procedure, to take an input and adjust its internal weights in such a way that the input data is best translated to the desired output. In this case, the network input was the six-dimensional pose vector produced by the CVS and the network output was the corresponding measurement error of the input pose vectors.

Finally, to validate the trained network's accuracy, it was tested with another set of input pose data of which the measurement error is already known. This data set can then be used to refine the network's training parameters.

The design process is iterative, where steps 2 and 3, and perhaps 1, can be repeated until a network which outputs satisfactory results is produced.

4.2.2 Network Training

Training a machine learning model is perhaps the most crucial aspect of the model design phase. Here, the nodes of the RBFNN are initialised and assigned a weighting. The weightings are then adjusted and optimised through a supervised training method called backprogagation, demonstrated in Figure 4.1. The number of radial bases can also be optimised to produce the best results, but in this case that number was preselected.

Backpropagation works as follows. The hidden nodes are assigned an initial weighting factor. The input training data is then fed to the network and its output is compared to the training set's output data. Based on this difference, the hidden nodes' weightings are adjusted in an attempt to minimise

the difference. The input data is then fed to the adjusted network again and its new output is compared to the output training data, where the weights are adjusted again. This process is repeated until the mean square error (MSE) of the output's deviation from the training data falls below a set threshold or converges to zero.

Factors that affect the network's accuracy are the number of nodes, or radial bases, in the hidden layer as well as the strictness of fit that the designer selects. The number of hidden nodes is a measure of the complexity of the data the network can process and also reflects its own complexity. However, when too many nodes are initialised there is a significant risk that the network will capture and attempt to model the noise and outlier data in a data set, instead of only characterising the underlying relationships between the input and output. This phenomenon is known as overfitting and can be observed when a network with a very small training MSE is exposed to new input data and the MSE is a few orders of magnitude larger than for the training data. Therefore, the validation phase is also an important aspect of network training to ensure that overfitting does not take place.

The strictness of fit measure determines to what extent the network will attempt to fit itself to the training data. Selecting the right parameter is important, since neglecting to replicate the training data adequately defeats the purpose of training the network to fit the data in the first place. However, if this parameter is too strict, the network will attempt to model the outlier and noisy data points as well, which is undesirable. Therefore, a fine balance needs to be found for the strictness of fit and the number of hidden nodes to produce a satisfactory network.

To train the network of this project, two sets of data was used; one set for training and another for validation. Both of these sets were taken from the data generated during the Vicon measurement test where the measurement accuracy of the CVS was determined. It includes both the pose data from the CVS and its corresponding measurement error. It was decided to use 300 training samples, giving 50 samples per dimension. The training data was selected to be uniformly distributed to place emphasis on the entire measurement spectrum. The uniformity of the data in each dimension was verified with the Chi-square (χ^2) goodness-of-fit test, with the hypothesis being that the set is uniformly distributed. As a rule of thumb, if the P-value for the χ^2 test falls below 5 %, then the hypothesis is statistically insignificant and must therefore be rejected. The training data was selected such that the P-value did not fall below 45 % for the data in each dimension.

Since the pose vector contains different measurement units (degrees and metres), the input and output training data sets were normalised to a range of $[-1, 1]$. All subsequent inputs to and outputs from the network must be normalised and denormalised with the same values that the training data was normalised with. To further accommodate the inherent differences between position and orientation data, two RBFNNs were trained: one to estimate the

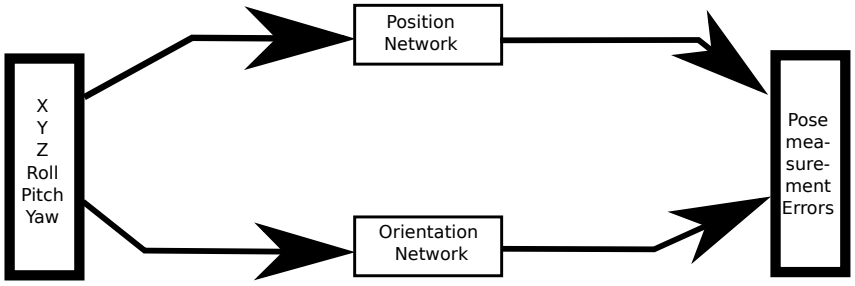


Figure 4.2: Figure showing the configuration for the dual RBFNN configuration.

position measurement error and another to estimate the orientation estimation error. Both networks take six-dimensional CVS pose measurements as inputs. Figure 4.2 shows the layout for the dual network configuration.

Matlab's[1] Neural Network toolbox[2] was used to train the RBFNNs and get the output for subsequent input data. The function prototype is given by

```
rbf = newrb(P,T,goal,spread,MN,DF).
```

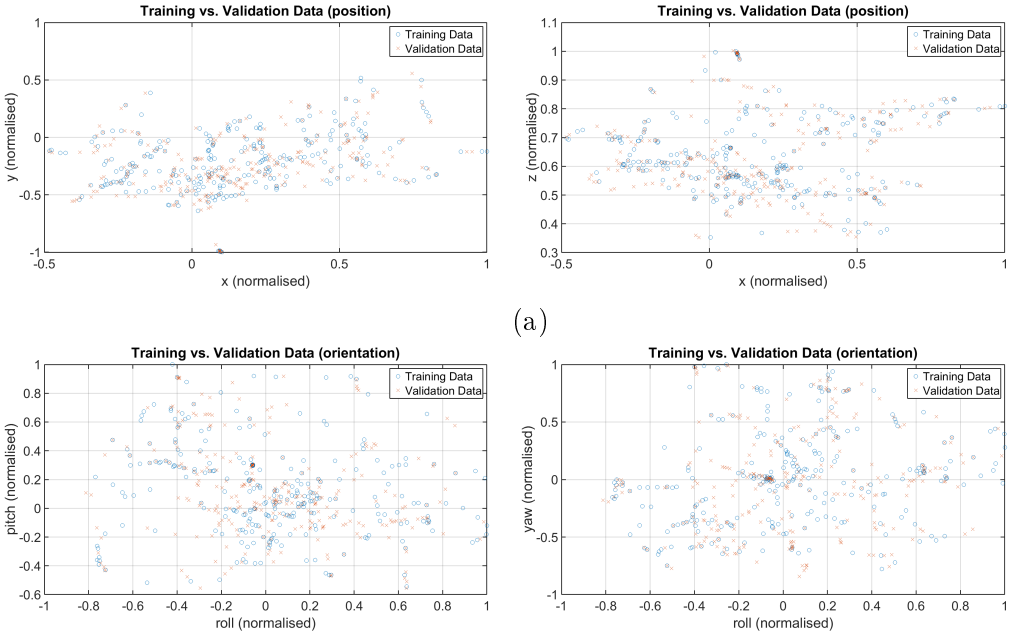
Here, P and T are the input and output training data matrices. The *goal* parameter is the error threshold for the training data and *spread* is the strictness of fit parameter, while *MN* and *DF* define the maximum number of hidden nodes and dictate the amount of nodes to increase between each training iteration. Adjusting the *spread* and *MN* parameters are the primary ways of manipulating the networks and produce more accurate outputs.

4.2.3 Model Validation

A model validation procedure was used to ensure that the RBFNNs were properly trained and that overfitting did not occur. The validation set comes from the same Vicon test that the training data was selected from. Another 300 randomly selected pose measurement samples were used for the validation data set. Given that the training data is uniformly distributed, the validation set should fall within the training data limits. Figure 4.3 displays the scatter plots for the position and orientation vectors for the training and validation sets and shows that they indeed do overlap to a large degree.

The measurement error was determined by feeding the validation input data into the trained networks and comparing their outputs with the validation error data. The MSEs between the networks' output and validation data were used as measures of accuracy. The validation set was used to tune the *spread* and *MN* training parameters, where the best combination would produce the smallest training and validation MSEs.

[1]Matlab v8.4.0.150421 (R2014)
[2]Neural Network Toolbox v8.2.1



(a)

(b)

Figure 4.3: Normalised scatter plots of the translation and rotation data points for the training and validation data sets. Note that the two data sets overlap. This is done on purpose to ensure that the RBFNNs do not extrapolate values when the validation set is used as input.

4.3 Results

With the training and validation data sets selected, network design could begin. Different combinations of training parameters for both networks were tested. It was found that the *spread* parameter had the largest influence over the networks' MSEs during training and validation. This may be because the input data (the orientation in particular) changes relatively quickly and contains a fair amount of noise. It was found that forcing the RBFNNs to go through all the points caused the MSE to rise in both networks when they were tested with the validation set, indicating that the networks were being overfitted. It was therefore decided to keep the *spread* parameter relatively small to allow the RBFNNs to ignore what they classify as outlier data and rather describe the general trend within the data.

Furthermore, changing the maximum number of hidden nodes also influenced the accuracy of the networks. Leaving the *MN* parameter to its default value will let the maximum number of hidden nodes go up to the number of training samples (300 nodes in this case), making the networks overly complex and slow. It was found that the networks are more accurate with both the training and validation data sets when the hidden nodes numbered between 20 % and 30 % the number of training samples.

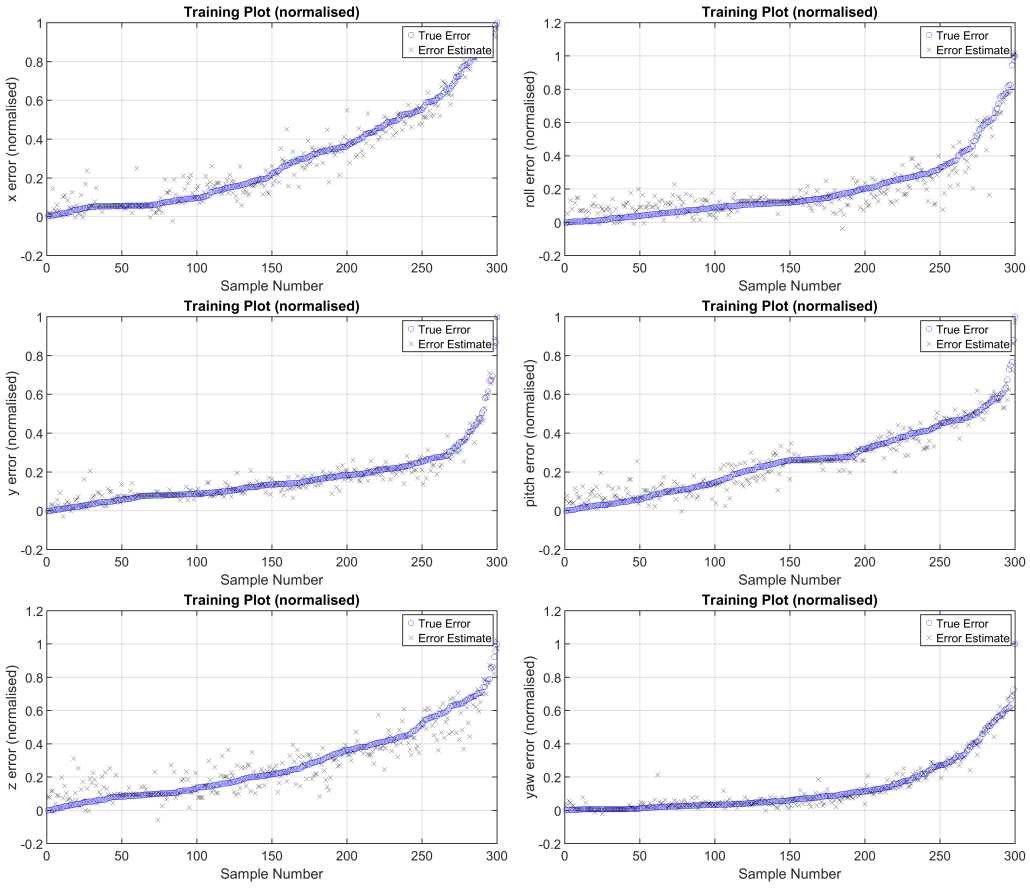
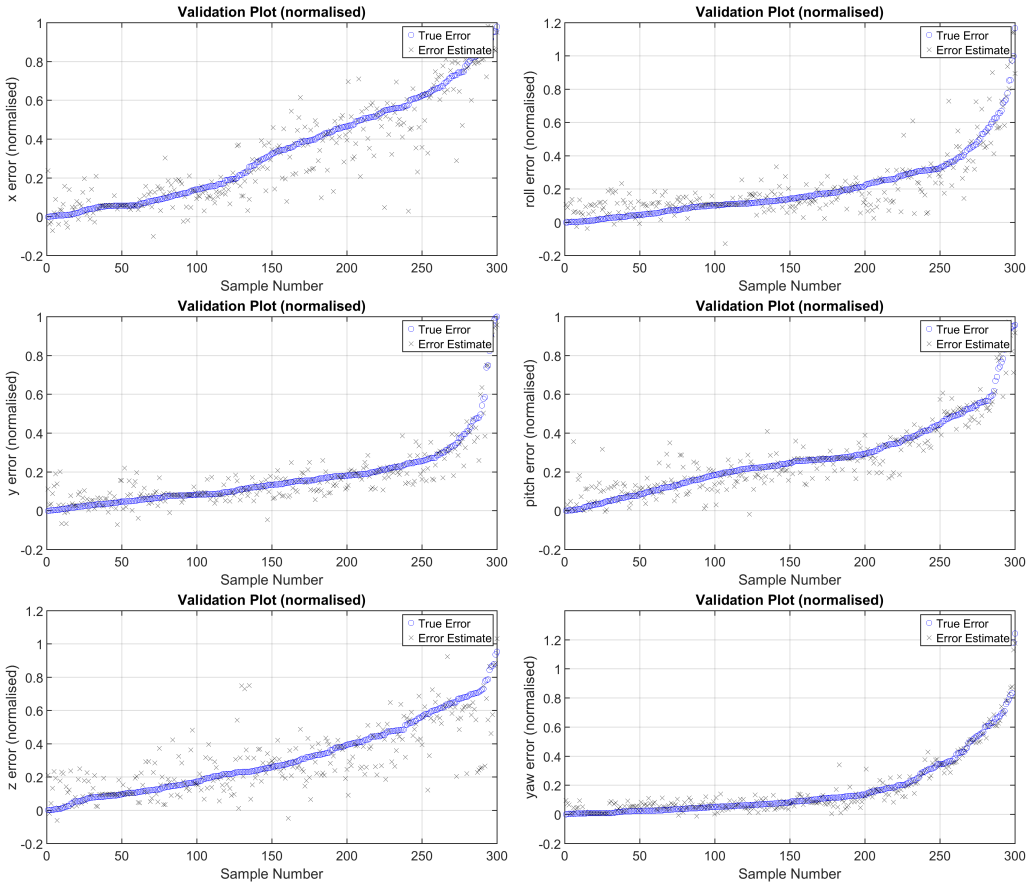


Figure 4.4: Plots of the output of the RBFNNs for the different dimensions when used with the training data set.

A training parameter combination that was found to work well for the translation network was a *spread* of 8×10^{-1} with 85 hidden nodes. This combination gives a normalised MSE of approximately 7.14×10^{-2} with the validation data set and 9.44×10^{-2} with the training set. For the orientation network it was found that a *spread* of 9×10^{-1} with 60 hidden nodes produces good results with a training MSE of 5.13×10^{-3} and a validation MSE of 6.1×10^{-3}. Leaving the networks to use their default *spread* and *MN* parameter values (1 and 300 respectively) produced a validation MSE 8 orders of magnitude larger than the training MSE. This displays the effect of overfitting well.

Figure 4.4 shows the results of the RBFNNs when tested with the training data set. It can be seen that the networks' estimates very closely resemble the training set's error, indicating that the networks were well-trained.

Figure 4.5 shows the networks' output when tested with the validation data set. Here it can be seen that the RBFNNs' error estimates closely follow the true error, with the position network's MSE being smaller in the validation case than for the training case. The errors in Figures 4.4 and 4.5 were normalised



(a) (b)

Figure 4.5: Plots of the output for the RBFNNs when used with the validation data set.

with the maximum error values given by

$$\Gamma = [76.2\,\text{mm}\ 54\,\text{mm}\ 56.5\,\text{mm}\ 11.78^\circ\ 17.93^\circ\ 38.74^\circ]$$

These plots show that the networks were well trained and produce accurate estimates of both the training and validation data sets.

4.4 Conclusion

In this chapter, two neural networks were designed to model the complex and dimensionally dependant pose measurement error of the CVS. Two radial basis function neural networks were selected to do this: the networks take a six-dimensional pose measurement vector from the CVS and outputs the corresponding measurement error for the input sample. The networks were trained and validated by two different data sets from the same measurement test and produce a validation mean square error of 7.14×10^{-2} 6.1×10^{-3} for

the position and orientation network respectively, indicating a good model fit for the data.

The trained networks are now ready to estimate the measurement error for the CVS with pose measurement data generated by a quadcopter in flight. This data can then be used to determine the pose estimation accuracy of a quadcopter in flight.

Chapter 5

Quadcopter Test

5.1 Introduction

The main objective of this research project is to determine the pose estimation accuracy of an airborne quadcopter in the outdoors. This can be done by comparing a quadcopter's on-board pose estimate with the pose measurement of an external measurement device.

In Chapter 3, a computer vision pose measurement system (CVS) was designed and tested and its pose measurement error was determined. It was found that the measurement data produced by the CVS is complex, high dimensional and interdimensionally dependant, as shown by the covariance matrix of Equation 3.14. This makes it complicated to estimate the measurement error for any given pose measurement vector produced by the CVS.

Neural networks excel at detecting underlying relationships within data if they are properly designed and trained. Radial basis function neural networks (RBFNN) have been shown to work well with noisy, complex and non-linear data and in Chapter 4, two RBFNNs were trained to to estimate the CVS's measurement error. The accuracy of the trained RBFNNs were verified by an additional data set and they are ready to be used in a quadcopter flight test.

The trained RBFNNs were used with data gathered from a test flight with a quadcopter. This chapter sets out to discuss the design and details of the flight tests that were conducted, including the testing procedure, flight conditions and data processing. The results are then presented and discussed.

5.2 Flight Test Design and Procedure

Measures were taken to ensure that the flight tests were performed where it would produce meaningful results without posing a safety risk to surrounding bystanders and equipment. This section discusses the planning that went into the flight test, as well as the equipment that was used. It also describes the test location and procedure, as well as the data processing involved.

Table 5.1: Suncopter specifications.

Specification	Amount	Detail
Battery	1	6-cell Li-Po
Controller	1	Pixhawk
Estimated full-charge flight time (no load)	N/A	45 minutes
Motors	4	T-Motor 4006
Rotor diameter	N/A	68.6 mm
Weight (with battery)	N/A	4.5 kg
Wingspan (tip to tip)	N/A	1.2 m

5.2.1 Flight Test Design

This subsection describes the flight test design, including the equipment, the location and testing conditions.

Equipment

The equipment and facilities that were required and used for the flight tests are given as follows:

- CVS camera and laptop.
- Suncopter quadcopter.
- Certified unmanned aerial vehicle pilot.
- Calibration board.
- Isolated test site.

The details of the CVS are discussed in Chapter 3. It consists of a camera, which captures the image data, and a laptop which records and processes the data. A certified unmanned aerial vehicle (UAV) pilot was used for safety reasons since the quadcopter was flown in close proximity to people and other equipment. The calibration board was used in conjunction with the CVS to provide the pose data of the quadcopter. Here, the calibration board is an ISO A3-sized[1], 6×5 square chessboard pattern calibration board.

The Suncopter quadcopter is a custom-built quadcopter for the Solar Thermal Energy Research Group's (STERG) research purposes. The Suncopter's physical specifications are given in Table 5.1. It is equipped with a Pixhawk controller which allows the Suncopter to be autonomously flown or controlled via remote control. The Pixhawk's specifications and sensors are listed in Tables 5.2 and 5.3.

[1] Paper size of 297 mm × 432 mm

Table 5.2: Pixhawk specifications.

Component	Specification
Manual control	Spektrum DX-7 Radio set
Memory	256 KB RAM, 2 MB Flash
Processor	168 MHz 32-bit STM32F427 Cortex M4 core
Telemetry	3DR 433 MHz Telemetry set

Table 5.3: Pixhawk sensor specifications.

Component
3DR uBlox GPS + Compass
Invensense MPU 6000 3-axis accelerometer / gyroscope
MEAS MS5611 barometer
ST Micro L3GD20H 16 bit gyroscope
ST Micro LSM303D 14 bit accelerometer / magnetometer

Location

The flight tests were conducted at the Mariendahl experimental farm owned and operated by Stellenbosch University (SU). It is also the location of STERG's Helio100 central receiver concentrating solar power (CSP) project. The exact location is in an empty grass field relatively far away from any roadways and buildings. The soil in the field is soft and uneven, which was taken into account during the tests and the data processing.

Flight Test Conditions and Layout

The flight tests were conducted on the 26^{th} of June 2015 at the Helio100 test site at Mariendahl. The weather conditions were close to ideal with very little wind, clear skies and a moderate temperature. The tests were conducted at 10 AM and there was still significant condensation present on the ground. See Appendix D for a more detailed weather and wind report recorded on site.

For the flight test, video data of the Suncopter with a calibration board attached to its underside was recorded, where the pose data was extracted off-line after the tests have been completed. The issue of turbulence introduced by a quadcopter flying too close to the ground was considered since it can negatively affect flight performance. A general rule of thumb to prevent ground effects from influencing a quadcopter's flight, as advised by Basson (2015), is to fly a quadcopter the length of one of its props from the ground. The CVS's camera was placed on top of a 2 m post which is significantly higher than the diameter of the Suncopter's props, thereby eliminating ground effects.

A certified pilot was employed during the test. His responsibility was to perform the manual piloting tasks, such as positioning the Suncopter and switching flight modes, as well as taking over piloting of the Suncopter and

safely land it in case flight stability is compromised and a catastrophic failure was likely to occur. This was done in order to guarantee the safety of the surrounding equipment and on-site personnel.

5.2.2 Flight Test Procedure

For each flight test, the Suncopter was manually positioned above the centre of the CVS's camera on the post and set to *loiter* mode. In this mode, the Suncopter attempts to hold the altitude, position and yaw angle it had when it was set to *loiter* mode while remaining stable, meaning that the Suncopter will attempt to hold a roll and pitch angle of 0°.

It has been established in Chapter 3 that the pose measurement data from the CVS is highly interdimensionally dependant, which implies that the accuracy of its pose measurements depend on the Suncopter's current pose relative to the CVS's camera. As a consequence, it was decided that several flight tests would be conducted, each with a slightly different distance or yaw angle relative to the CVS's camera.

Distances of 1 m and 2 m from the camera were used. During preliminary flight tests, it was found that with distances greater than 2 m from the camera, the CVS's camera started losing sight of the corners on the calibration board and struggled to detect and extract the corner coordinate data from the calibration board, making it impossible to perform pose estimation. Furthermore, given that a quadcopter is symmetric about both of its axes, the yaw angles for the flight tests were set to 0°, 22.5° and 45°.

Two sets of data were recorded during the flight test: one from the CVS and another from the on-board sensor suite of the Suncopter which provided position and orientation data. These two data sets eventually allowed the on-board pose estimation accuracy of the Suncopter to be determined.

5.2.3 Data Processing

After the video data of the Suncopter with the calibration board attached was recorded, data processing could begin. There are four different processing phases. They are the pose extraction from the video data, synchronising the data to a common time frame, data recentering around a common centre, and finally determining the pose estimation accuracy of the Suncopter. Each of these phases are discussed in this section.

Pose Data Extraction

The first step in the data processing phase is to extract the pose data of the calibration board and, by extension, the Suncopter to which it was attached. This was done by using the process discussed in Chapter 2, where OpenCV's

chessboard corner detector and PnP problem solver were used to extract six-dimensional pose information of a chessboard pattern calibration board. This process is fairly automated. However, the output from the PnP solver still required some conditioning.

The coordinate system for the corner detector and PnP solver used is specified in square units, i.e. a 5×6 square chessboard would have dimensions of 5×6 units. This required the position vector to be transformed back to SI units by multiplying the PnP solver's output by 0.05, since each square on the chessboard is $5\,\text{cm} \times 5\,\text{cm}$. Similarly, the orientation vector is output in radians, which was then converted to degrees.

Data Synchronisation

The CVS's camera captures video data at 30 frames per second, while the Pixhawk on the Suncopter sends data to the ground control station at a frequency of 200 Hz, making it necessary to change the time scale of either data set so they match one another. The Pixhawk's data packets contain various parameters, such as the battery voltage and altitude, but only the position and orientation data sets are of interest here. However, this data is not sent with every data package coming from the Pixhawk and do not necessarily arrive together either. It was therefore necessary to synchronise the position and orientation data received from the Pixhawk as well.

Since the CVS's data was recorded at a reliable, steady rate of 30 Hz, a zero-order-hold was applied to the CVS's data to synchronise it with the quadcopter's data set. This was done by keeping a value in the CVS data set constant until the next timestamp in the quadcopter data set is reached. At that point, the CVS's data was updated to value it had at that time and was held constant until the next timestamp was reached. This process simultaneously downsampled and synchronised the CVS data.

Both data sets were recorded on the same laptop and therefore share a common time source. However, their start and end points differ. To synchronise these points, the creation time of the video file was used as a start point and the video was cut at the quadcopter's final data packet's timestamp.

Data Centering

After the pose data has been extracted from the video data, the CVS and Suncopter's pose estimates were recentred around a common axis system. The CVS's axis system is centred around its camera centre, while the Suncopter's pose data is centred around the point from where it is launched. Therefore, to be able to compare the two sets of data, the constant offset vector between the axis systems was determined. The axis systems and offset vector is shown in Figure 5.1.

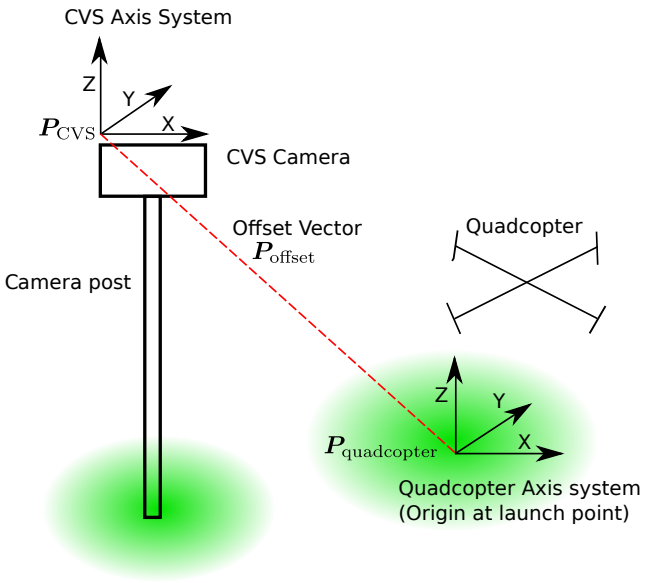


Figure 5.1: Schematic of the test flight, the quadcopter and CVS's respective axis systems and the offset vector between them.

Furthermore, any angular offset for the CVS camera or the Suncopter due to uneven ground conditions or imperfect camera placement is also included in the offset vector. This makes the offset vector a six-dimensional position and orientation vector. The offset vector is a constant value and is related to the quadcopter and CVS's pose estimate by the relation given in Equation 5.1, where $\boldsymbol{P}$ is a six-dimensional pose vector.

$$\boldsymbol{P}_{\text{quadcopter}} = \boldsymbol{P}_{\text{CVS}} + \boldsymbol{P}_{\text{offset}} \tag{5.1}$$

Here, $\boldsymbol{P}_{\text{offset}}$ is the unknown term. The pose data contained within the $\boldsymbol{P}_{\text{quadcopter}}$ and $\boldsymbol{P}_{\text{CVS}}$ terms contain noise in each sample, which requires an optimisation procedure be used to find an offset vector that best relates the two sets of pose data. For the optimisation procedure, the optimal offset vector, $\hat{\boldsymbol{P}}_{\text{offset}}$, and error term of Equation 5.2 was minimised, where the error vector, $\boldsymbol{e}$, is a function of $\boldsymbol{P}_{\text{offset}}$ and M is the number of samples in the data set. $\boldsymbol{e}$ is given by Equation 5.3.

$$\hat{\boldsymbol{P}}_{\text{offset}} = \min_{\boldsymbol{P}_{\text{offset}}} \sqrt{\sum_{i=1}^{M} (\boldsymbol{e}_i(\boldsymbol{P}_{\text{offset}}))^2} \tag{5.2}$$

$$\boldsymbol{e}_i(\boldsymbol{P}_{\text{offset}}) = \boldsymbol{P}_{\text{CVS},i} + \boldsymbol{P}_{\text{offset}} - \boldsymbol{P}_{\text{quadcopter},i} \tag{5.3}$$

The minimisation was done using the *minimize()* function from the SciPy[2] library for the Python language, which makes use of the *BFGS* quasi-Newton method described by Nocedal and Wright (2006).

[2]SciPy scientific tools library v0.13.3

Determining the Pose Estimation Error

With the two data sets now directly relatable, the pose estimation error of a quadcopter in flight can now be determined by using the RBFNNs discussed in Chapter 4 and the two sets of pose data recorded during the flight tests.

The input to the networks are the CVS's recorded pose data. The networks then output the expected measurement error for every sample, allowing a measurement error band to be drawn around the CVS's measurement data. This error band can then be used to determine if the quadcopter or the CVS's pose estimates are more accurate: when the quadcopter's estimate falls within the error band, it indicates that the quadcopter's pose estimate is more accurate than the CV's for that sample, while the opposite holds true if it falls outside the error band. It can then be determined which of the two systems produce, on average, the most accurate pose measurements. However, if it is found that the quadcopter's pose estimate is more accurate, one might be liable to ask if the CVS is still necessary.

Up to this point, any measure of a quadcopter's pose estimation error has been unavailable for outdoor quadcopters. This means that even if it is found that the quadcopter's pose estimate is more accurate than the CVS's, the CVS pose data will still provide a worst-case measure of the quadcopter's pose estimate which was not available previously.

The outcome of this comparison may lead to one of the following two outcomes: if the quadcopter's pose estimate is more accurate than the CVS's, the pose measurement error of the CVS can be taken as a quadcopter's worst-case pose measurement error, where the quadcopter's pose estimate will always be at least as accurate as, but likely better than the CVS's. Conversely, if it is found that the CVS produces better pose measures, then the quadcopter's pose estimation error will be given by the difference between the CVS and quadcopter's pose measurement data.

5.3 Results

The results for the optimum offset vector, $\hat{\boldsymbol{P}}_{\text{offset}}$, and the pose estimation error is presented and discussed in this section.

5.3.1 Offset Vector

Using the procedure described in Section 5.2.3, the optimal offset vector relating the quadcopter and CVS's axis systems is given by[3]

$$\hat{\boldsymbol{P}}_{\text{offset}} = \left[-6.61\,\text{m}\ -0.834\,\text{m}\ -4.21\,\text{m}\ 8.47^\circ\ -1.623^\circ\ -183^\circ\right]^T. \tag{5.4}$$

[3]Different starting points for the optimisation procedure were used to ensure that the procedure did not get stuck in a local minimum.

(a)

(b)

Figure 5.2: Scatter plots showing that the flight test data falls within the training limits.

The position values roughly coincide with the conditions that were recorded on site: the Suncopter was launched some distance from the CVS equipment for safety reasons. Furthermore, due to uneven ground conditions at the testing site, the launch site was also slanted and located below the post on which the CVS's camera was placed. Thus, from a sanity check point of view, the offset that the optimisation procedure produced seems to be the correct one.

5.3.2 Pose Estimation Error

Several different flight tests were conducted with different flight conditions. However, for convenience, only the results from the 1 m altitude and 0° yaw case are presented and discussed. There is nothing special about this case and the data results from all the flight tests produce roughly the same outcomes.

Before the flight test data was fed to the RBFNNs, it was first checked that the test data fell within the training data limits. This was done to ensure that the RBFNNs were not exposed to data they were not trained for and produce inaccurate results. Figure 5.2 shows scatter plots for the test and training data dimensions and shows that the test data fell comfortably within the training data set's limits.

The results for the flight test are presented in Figures 5.3 to 5.8. Here, six figures are given which plot the different pose dimensions. Each plot contains the Suncopter's pose estimate, as well as the expected measurement error

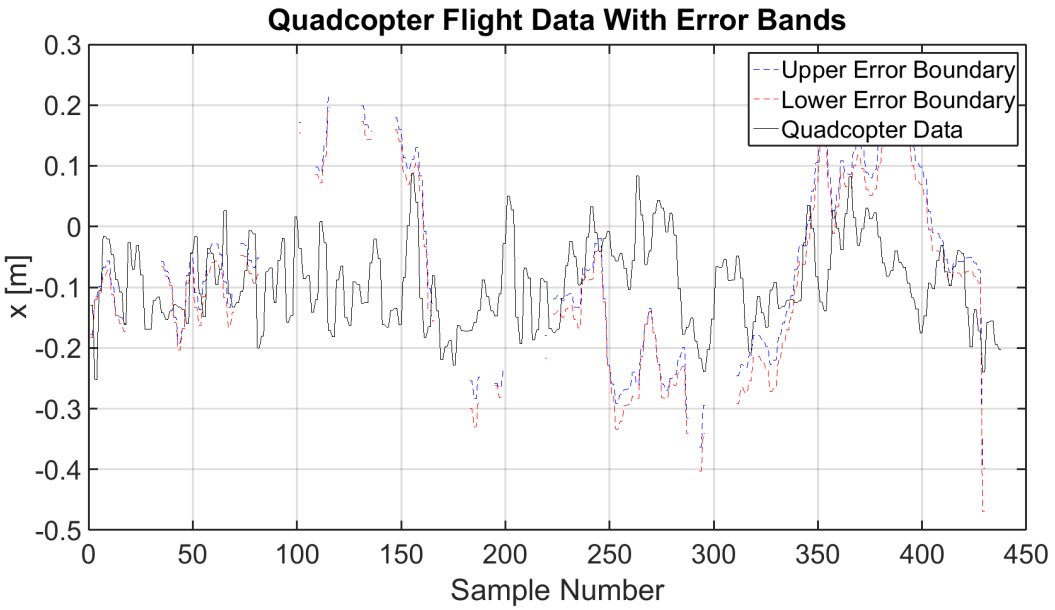


Figure 5.3: Results of the flight test, comparing the CVS and quadcopter's measurements in the x dimension.

margin for the CVS as given by the RBFNNs. It can be observed from the plots for the z and *yaw* dimensions that they consistently hover around the flight configuration's set point of 1 m and 0°, providing a degree of sanity to the measurements.

An important aspect of the plots in Figures 5.3 to 5.8 to take note of is that sections in the CVS error estimate plots are missing. The reason for this is that during the video recording, the calibration board went out of the CVS camera's field of view or it could not detect enough of the corners due to bad lighting conditions or the board being to far from the camera. These effects were somewhat remedied by adjusting the pose extractor to use an adaptive threshold filter on the video data and can be permanently fixed by flying the quadcopter closer to the camera or ensuring that the board is well and evenly lit throughout the test.

From Figures 5.3 to 5.8 it can be seen that all of the quadcopter's position estimates, as well as its roll and pitch estimates, fall largely outside the CVS's error boundaries given by the RBFNNs. This implies that the CVS's position, roll and pitch measurements are more accurate than the quadcopter's. Both the position and orientation RBFNNs were well trained and produced small training and validation MSEs, providing a high level of confidence that the true position and orientation of the quadcopter lies within the CVS's error boundaries. The quadcopter's position, roll and pitch estimation accuracy can consequently be determined by comparing the two sets of measurement data.

The quadcopter's *yaw* estimates fall inside the CVS's error boundaries for the most part, implying that the quadcopter's estimates are more accurate

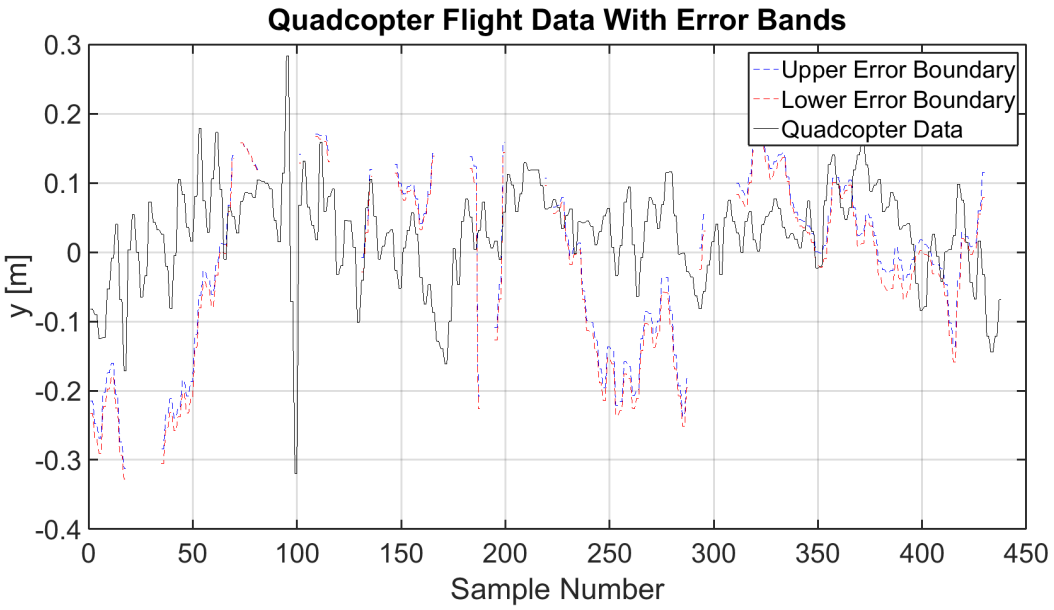


Figure 5.4: Results of the flight test, comparing the CVS and quadcopter's measurements in the y dimension.

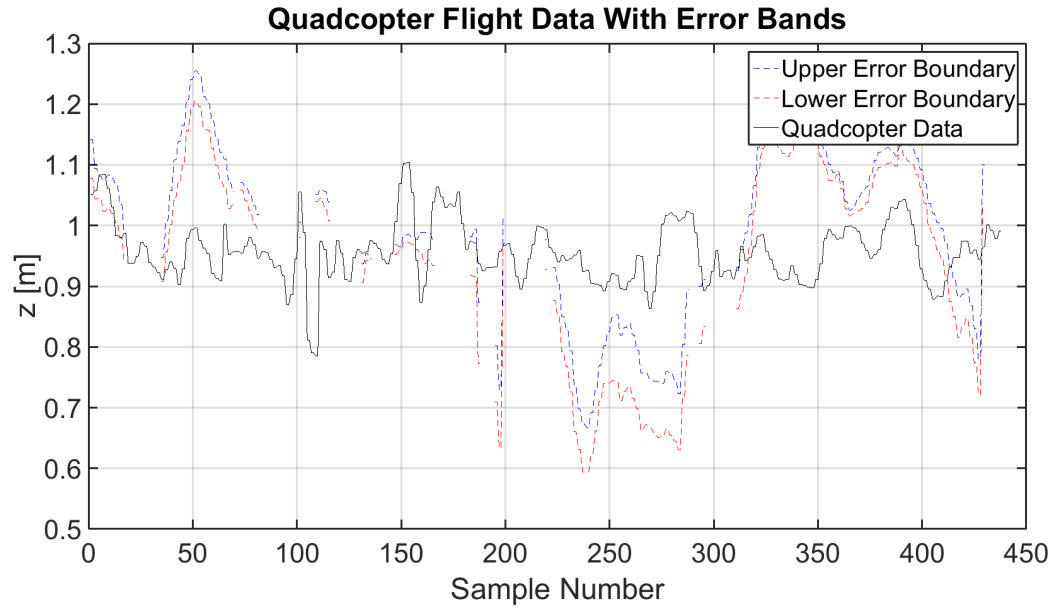


Figure 5.5: Results of the flight test, comparing the CVS and quadcopter's measurements in the z dimension.

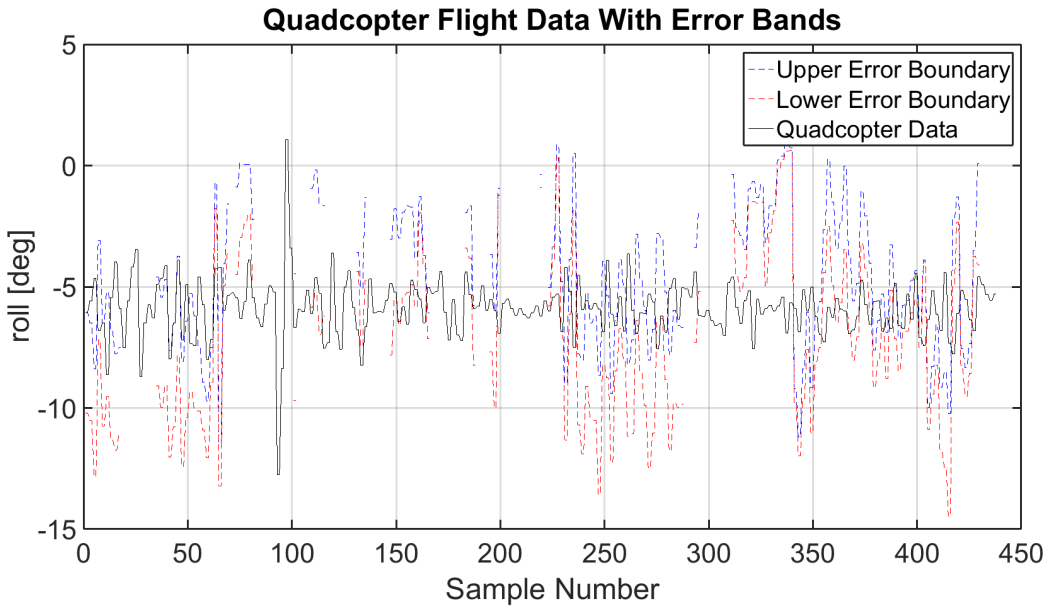


Figure 5.6: Results of the flight test, comparing the CVS and quadcopter's measurements in the *roll* dimension.

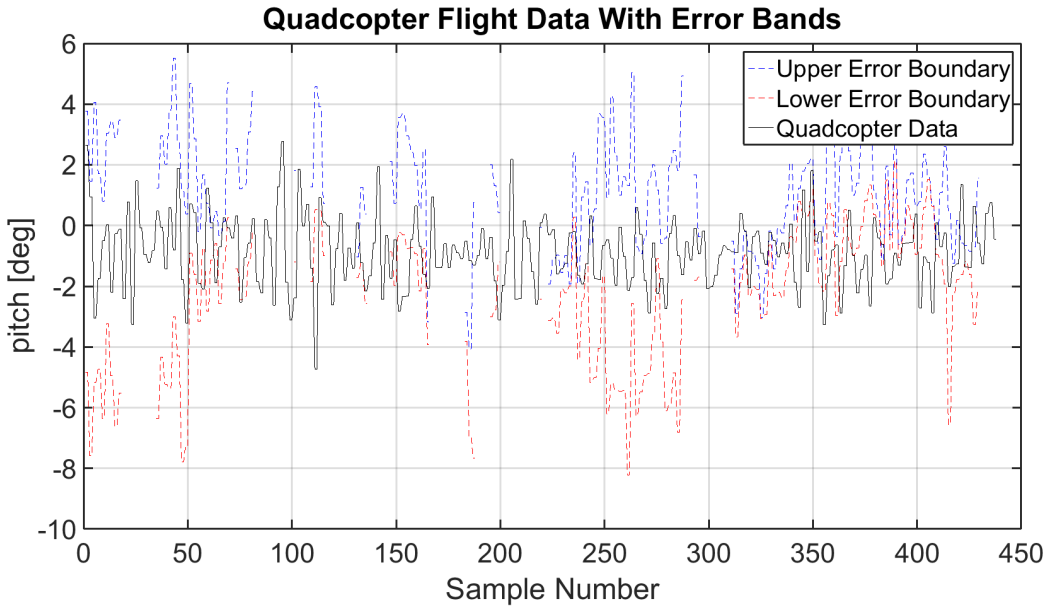


Figure 5.7: Results of the flight test, comparing the CVS and quadcopter's measurements in the *pitch* dimension.

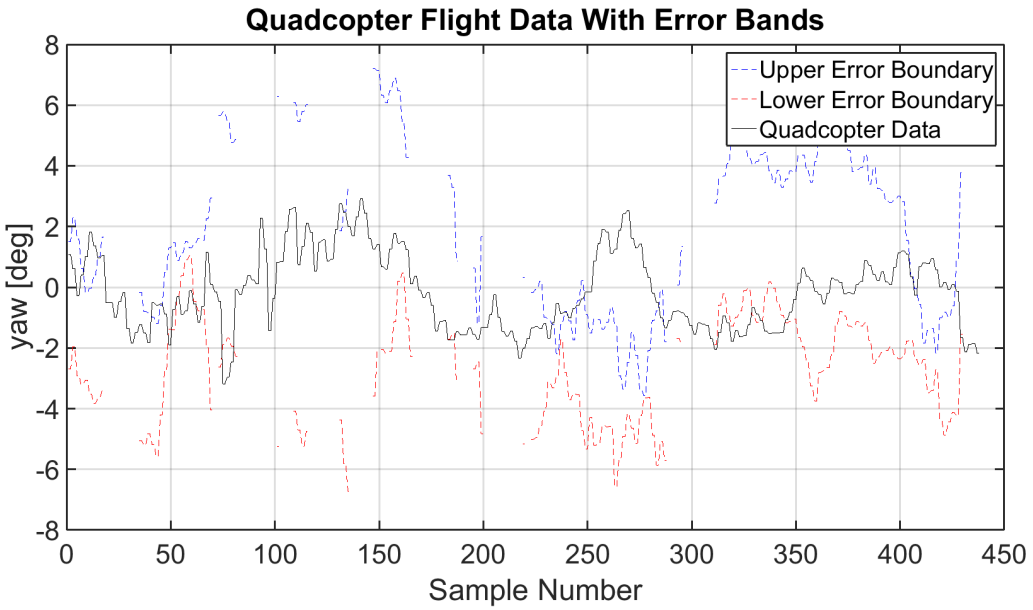


Figure 5.8: Results of the flight test, comparing the CVS and quadcopter's measurements in the *yaw* dimension.

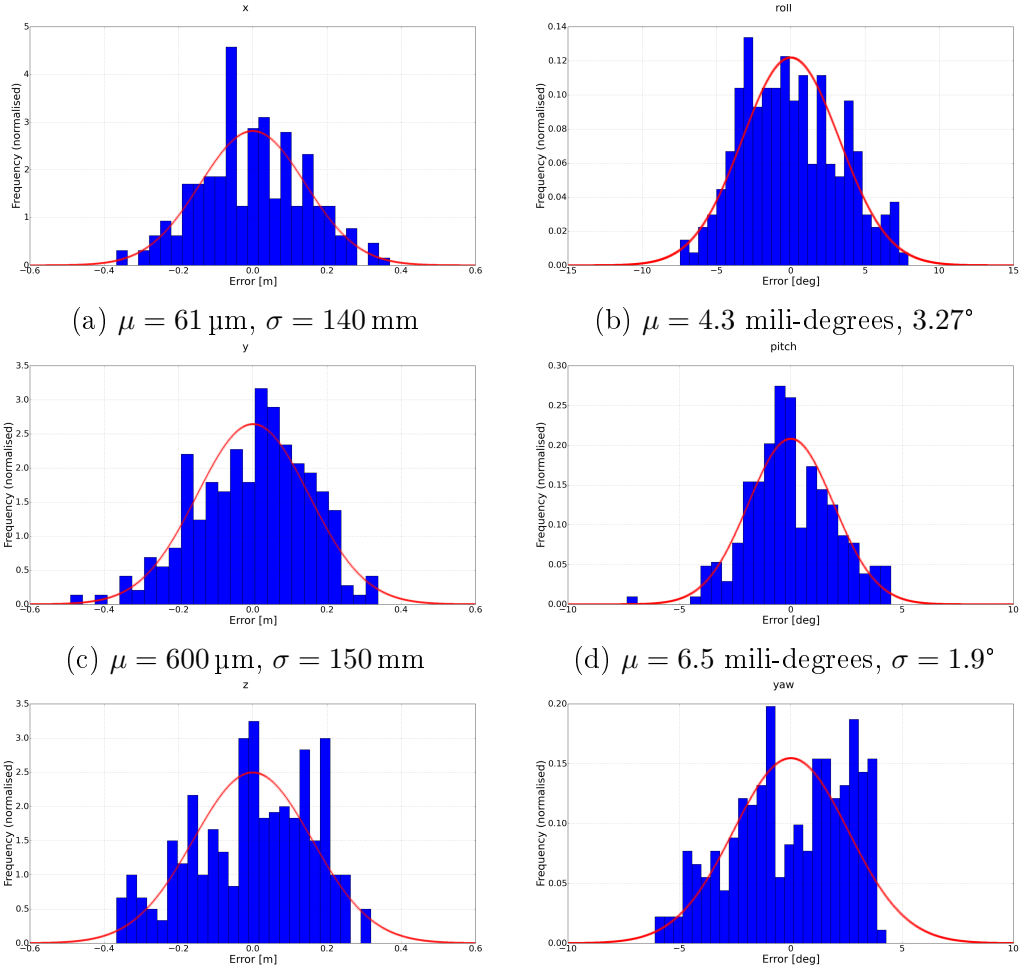
than the CVS's. This is not unexpected, since Figure 3.9 has shown that the CVS performs the worst in the *yaw* dimension. In this case, the CVS's error boundary can be used as a worst-case pose estimate error for the quadcopter, since the quadcopter's estimate will not be worse than the CVS's.

Figure 5.9 shows a set of frequency histogram plots; one for each dimension. Each plot shows the frequency of the deviation between the CVS and quadcopter measurements.

It can be seen from Figure 5.9 that the error deviation, i.e. the difference between the CVS and quadcopter's measurements, are normally distributed about a mean of zero. All three of the position dimensions display similar levels of deviation from the CVS's measurements, with a 1-sigma level of approximately 150 mm for all three. The pitch and yaw dimensions have a 1-sigma level of deviation of 3.27° and 1.9° respectively. These standard deviations give an indication of the quadcopter's pose estimation accuracy.

5.4 Conclusion

In this chapter, the CVS and RBFNNs were used to measure the pose of an airborne quadcopter in the outdoors and to determine the quadcopter's pose estimation accuracy. It was found that the quadcopter's position estimation accuracy in all three dimensions is approximately 150 mm and 3.27° and 1.9° for the *roll* and *pitch* dimensions respectively. However, in the *yaw* dimension the quadcopter is more accurate. In this case, the measure of accuracy that can



(a) $\mu = 61\,\mu\text{m}$, $\sigma = 140\,\text{mm}$

(b) $\mu = 4.3$ mili-degrees, $3.27°$

(c) $\mu = 600\,\mu\text{m}$, $\sigma = 150\,\text{mm}$

(d) $\mu = 6.5$ mili-degrees, $\sigma = 1.9°$

(e) $\mu = -105\,\mu\text{m}$, $\sigma = 160\,\text{mm}$

(f) $\mu = 3.4$ mili-degrees, $\sigma = 2.6°$

Figure 5.9: Set of frequency histograms displaying the quadcopter's pose estimate deviation from the CVS's measurements. Even though it has no bearing on the results, the *yaw* dimension's plot is included here for the sake of completeness.

be used is the error boundary given by the orientation RBFNN. This provides a worst-case error estimate that can be implemented.

Chapter 6

Conclusion

6.1 Introduction

This research project set out to accomplish two things: find a reliable and relatively cheap method if measuring the pose of an object in the outdoors and then use that system to determine the pose estimation accuracy of an unmanned aerial vehicle (UAV) quadcopter in flight in the outdoors. The pose refers to a six-dimensional vector describing an object's position and orientation.

Previous chapters have covered each aspect of the computer vision system's design, testing and implementation process, as well as live testing with a quadcopter. This chapter provides a brief summary of the project, as well as the key findings and results. The significance of these findings to the existing body of knowledge is also discussed. Finally, the shortcomings of the findings, as well as potential improvements and future work, are discussed.

6.2 Thesis Summary

Chapter 1 started the document off with an introduction to the problem, motivated the reason why a new outdoor measurement system needed to be developed and what the potential benefits are to knowing how accurately a UAV can estimate its pose.

A review of the current body of knowledge pertaining to the relevant aspects of this research project was given in Chapter 2. It was found that a significant amount of research has already been done in stabilising a UAV and having it hold its position in the air using different control strategies, but thus far there is no literature discussing how well an outdoor quadcopter can estimate its pose. Furthermore, it was established that there are well-understood and widely researched computer vision techniques and libraries available that can extract the pose data of an object from an image. Finally, this chapter includes a brief review of the different machine learning techniques available

that can be trained to make accurate measurement error predictions for any given pose measurement vector. The different techniques and their strengths and limitations are discussed.

In Chapter 3, all of the design aspects of the computer vision system (CVS) was discussed, which include the hardware and software design, as well as the measurement accuracy test and verification process. It was found that the complex nature of the CVS's measurement error necessitates a measurement error estimation model to predict what the measurement error would be for any input pose measurement vector from the CVS.

Chapter 4 set out to discuss the use of radial basis function neural networks (RBFNN) to estimate the measurement error for any pose measurement vector produced by the CVS. It discusses the training process and how the trained networks' accuracies were verified. Two RBFNNs were trained (one to handle the position dimensions and another for the orientation) which produce satisfactory levels of error with their estimates and can be used with new pose measurement data.

Finally, Chapter 5 is dedicated to discussing how the CVS and the RBFNNs were used in a flight test with a real quadcopter. A number of flight tests were performed with the Solar Thermal Energy Research Group's (STERG) Suncopter quadcopter platform. During these tests, both the CVS and the Suncopter recorded the quadcopter's pose information. The quadcopter's pose measurement accuracy was determined for all the dimensions, except for the *yaw* dimension. In this case it was found that the quadcopter produces more accurate measurements than the CVS does. The CVS's error boundaries can still be used as a worst-case measure of error for the quadcopter's yaw estimates.

6.3 Findings and Contributions to Body of Knowledge

The objectives of this research thesis are discussed in Chapter 1. They are stated here again for convenience and are as follows:

- Design and implement a relatively cheap computer vision pose measurement system.
- Determine the measurement accuracy of the pose measurement system.
- Use the computer vision system to determine the pose estimation accuracy of a demonstration quadcopter in flight.

For the first objective, a computer vision-based system was made which uses a single camera to record video data of a calibration object and the OpenCV library to extract six-dimensional pose data of the object. Such

a system was required since indoor measurement tools cannot be used to determine an outdoor quadcopter's pose estimation error. It was decided to investigate whether a computer vision system is a viable outdoor measurement solution since existing outdoor measurement systems are very expensive and it was desirable to investigate whether a cheaper alternative could provide the same functionality. The computer vision system is fairly simple, cheap, easy to use and set up. Preliminary tests have shown that its estimates were within the ballpark, however, its pose measurement accuracy had to be accurately determined before it could be used to perform any pose measurements.

The measurement system's pose accuracy was determined by comparing it with the ground-truth pose measurements of a state-of-the-art Vicon motion tracking system. The measurement error is fairly complex and high-dimensional and the measurement test has shown that it is highly interdimensionally dependant (as shown by the covariance matrix in Equation 3.14 and the plots in Figure 3.16). This means that the CVS's pose measurement accuracy is dependant on the calibration object's pose relative to the CVS's camera. Therefore, another method of predicting the CVS's pose measurement error for any given pose measurement vector was required.

It was decided that two RBFNNs would be used to perform the measurement error estimation: one for the position dimensions and another for the orientation. This was done to accomodate the inherent differences between position and orientation data. A RBFNN network type was selected based on its proven ability to work well with noisy input data and detect non-linear relationships between the input dimensions. The RBFNNs were trained with a uniformly distributed data set and validated with randomly selected data points. Both data sets were selected from the same Vicon test. The networks produce acceptably small mean square errors for the validation set. It was found that the RBFNNs perform better in the position dimensions than in the orientation dimensions, with a possible reason being the large amount of noise within the orientation data.

Finally, the CVS and RBFNNs were used together to determine the pose estimation accuracy of a quadcopter in flight. This was done by performing flight tests with a real quadcopter in the outdoors. The results of the test flight gave an indication of the pose estimation accuracy of a quadcopter. It can be seen from Figure 5.9 that the position dimensions have a standard deviation from the CVS's measurements of approximately 150 mm for the position dimensions and 3.27° in the *roll* dimension and 1.9° in the *pitch* dimension. The quadcopter's yaw estimate was found to be more accurate than the CVS's. However, the CVS's error boundary can be used as an expected error measure in this case. These results can be incorporated into a quadcopter to improve its control strategy and evaluate the performance of current controllers.

The work done for this project have been presented in part at the SolarPACES 2015 conference in Cape Town. A proceedings article has also been accepted and is pending publication (Lock *et al.*, 2015).

6.4 Shortcomings and Future Work

The main research objectives and goals that have been set for this thesis have been achieved. However, there are some issues that have been encountered during the project. There is also potential to take this research further to improve and implement the results. These shortcomings and a few potential avenues of further research are mentioned and discussed here.

6.4.1 Improved Measurement System Accuracy

In this project a computer vision-based pose measurement system (CVS) was designed and tested. It was found that it produces fairly accurate measurement results and, in conjunction with a neural network measurement error estimator, can be used to measure the pose of a chessboard pattern calibration board. This is a first design iteration and there are several improvements that can be made to improve the system's accuracy.

One change to the CVS which can be implemented is to use high-definition (HD) video data. It is expected that using an HD camera would improve the CVS's pose measurements, since there are more pixels available and the CVS would be able to determine the chessboard's corner coordinates with a finer degree of accuracy. In this project, low resolution (640×480 pixels) video data was used. The reason for using the lower resolution during the system design phase is that the pose estimation for HD video data took approximately 3.5 times longer than for the low resolution video data (approximately 6.5 hours). This is due to the higher number of pixels in the HD video (3 times more pixels in this case). Due to this long processing time and since it was anticipated that the same program will be run many times throughout the CVS's design process, it was opted to use the low resolution video data instead. The HD upgrade is ready to be implemented, since the CVS's current camera has an HD resolution option of 720×1280 pixels.

Furthermore, different markers for the CVS can be implemented. For example, attaching a chessboard to the underside of a quadcopter influences its dynamics and disrupt the airflow around it. Therefore, if a set of markers could be placed on the underside of the quadcopter, it would provide the same level of accuracy the current system has without influencing the quadcopter's dynamics, provided the same number of markers are used as there are corners on the chessboard. Also, instead of a two-dimensional planar calibration pattern, a three-dimensional calibration object can be used. According to Medioni and Kang (2004), using a precisely constructed three-dimensional rig would provide more accurate calibration and pose measurement results.

6.4.2 Gaps in the Pose Measurement Data

It was observed with the flight test data that the CVS's pose measurements contain gaps where no data was recorded. Inspection of the video material showed that for significant sections of the video, OpenCV's corner detection algorithm could not detect all of the expected corners on the calibration board.

There are a few reasons why this may have occurred. One is that the board was simply too far away and the detector could not get a good enough estimate of the corners. Another is that the uneven lighting conditions in the field test led to the contrast not being great enough to highlight the difference between the white and black tiles on the board. This was somewhat remedied by using an adaptive threshold filter in the corner detector. However, there are still some gaps within the data set.

A suggestion to permanently fix this issue would be to ensure that the calibration board be evenly lit throughout the test. One possibility would be to perform the test while using spot lights to control the lighting level on the calibration board and ensure that the board stays well and evenly lit.

6.4.3 Improved Sensor Hardware

One avenue of improvement, unrelated to the CVS, which may increase the pose measurement accuracy of a quadcopter significantly, is to improve the sensor hardware it is equipped with. More specifically, if a more accurate GPS sensor can be used, it should show a significant improvement in a quadcopter's localisation results. An improved state estimator and sensor fusion solution may also produce significantly better results.

Currently, the Suncopter comes equipped with a standard uBlox GPS module, which has an expected accuracy of within 3 m (approximately the normal level of accuracy for a standard GPS module). However, recent improvements in the field of differential GPS technology, especially real-time kinematic (RTK) GPS's, have led to smaller, more accurate GPS units becoming available for use with a quadcopter. There already has been some development done in implementing the Piksi RTK GPS in the Pixhawk controller. The Piksi GPS has a reported accuracy level of within a few centimetres, which is a significant improvement over the traditional GPS modules.

Other sensor improvements can be made, such as a more accurate magnetometer, inertial measurement unit (IMU), and so on. However, it is not expected that these sensors will have as significant an effect on the quadcopter's localisation ability as adding a differential GPS unit would.

6.4.4 Implement Results

The pose estimation accuracy of a quadcopter has been determined in Chapter 5 and is ready to be implemented in a real drone. These errors can be

incorporated into a quadcopter's controller as an error term which will improve the controllers performance and safety margins.

Implementing these results will also help several other industries. For example, STERG are looking into using a quadcopter to calibrate heliostat mirrors on a concentrating solar power plant. In this context it is very important to have accurate position data available, or at least know what the expected measurement error is so that it can be taken into account during calibration.

6.5 Conclusion

This project has designed and tested a computer vision-based pose measurement system (CVS) and used to determine the pose estimation accuracy of an outdoor quadcopter in flight. Despite being in its first design iteration, the CVS performed well and produces accurate measurement results which were used to determine a quadcopter's pose estimation accuracy. This accuracy measure was not available previously and can be implemented into a real quadcopter to evaluate and improve its performance. This would be a great advancement into making quadcopters safer and make them more attractive to governments and industry alike.

It is hoped that these findings will help in improving quadcopters and help the technology live up to its massive potential.

Appendices

Appendix A

Pixhawk Specifications

Pixhawk specifications as taken from Arducopter (2015).

- Processor
  - 32-bit ARM Cortex M4 core with FPU
  - 168MHz/256KB RAM/2MB Flash
  - 32-bit failsafe co-processor
- Sensors
  - MPU6000 as main gyroscope and accelerometer
  - ST Micro 16-bit gyroscope
  - ST Micro 14-bit accelerometer/compass (magnetometer)
  - MEAS barometer
- Power
  - Ideal diode controller with automatic failover
  - Servo rail high power (7V) and high-current ready
  - All peripheral outputs over-current protected, all inputs ESC protected
- Interfaces
  - 5x UART serial ports. 1x high-power cable and 2x with HW flow control
  - Spektrum DSM/DSM2/DSM-X satellite input
  - Futuba S.BUS input (output not yet implemented)
  - PPM sum signal
  - RSSI (PWN or voltage) input

  - I2C, SPI, 2x CAN, USB
  - 3.3 and 6.6 ADC inputs
- Dimensions
  - Weight 38 g
  - Width 50 mm
  - Height 15.5 mm
  - Length 81.5 mm

Appendix B

Lifecam HD-5000 Specifications

Microsoft®
LifeCam
HD-5000

Version Information	
Product Name	Microsoft® LifeCam HD-5000
Product Version	Microsoft LifeCam HD-5000
Webcam Version	Microsoft LifeCam HD-5000
Product Dimensions	
Webcam Length	1.49 inches (37.8 millimeters)
Webcam Width	1.61 inches (40.8 millimeters)
Webcam Depth/Height	4.29 inches (109 millimeters)
Webcam Weight	3.40 ounces (96.5 grams)
Webcam Cable Length	72.0 inches (1829 millimeters)
Compatibility and Localization	
Interface	High-speed USB compatible with the USB 2.0 specification
Operating Systems	Microsoft Windows® 7, Windows Vista®, and Windows XP with Service Pack 3 (SP3) excluding Windows XP Pro 64-bit
Top-line System Requirements	Requires a PC that meets the requirements for and has installed one of these operating systems: • Microsoft Windows 7, Windows Vista, or Windows XP with Service Pack 3 (SP3) excluding Windows XP Pro 64-bit • Intel Dual Core 1.6 GHz (Intel Dual Core 3.0 GHz recommended) • 1 GB of RAM (2 GB of RAM recommended) • Video Card with 1280 X 720 pixels or higher • 1.5 GB of hard drive space • Windows-compatible speakers or headphones • USB 2.0 You must accept License Terms for software download. Please download the latest available software version for your OS/Hardware combination Internet access may be required for certain features. Local and/or long-distance telephone toll charges may apply. Software download required for full functionality of all features. Internet functions (post to Windows Live™ Spaces, send in e-mail, video calls), also require: Internet Explorer® 6/7/8 browser software required for installation; 25 MB hard drive space typically required (users can maintain other default Web browsers after installation)
Compatibility Logos	• Compatible with Microsoft Windows 7 • Certified High-Speed USB logo
Software Localization	Microsoft LifeCam software version 3.2 may be installed in Simplified Chinese, Traditional Chinese, English, French, German, Italian, Japanese, Korean, Brazilian Portuguese, Iberian Portuguese, Russian, or Spanish. If available, standard setup will install the software in the default OS language. Otherwise, the English language version will be installed.
Windows Live™ Integration Features	
Video Conversation Feature	Windows Live Call button delivers one touch access to video conversation.
Call Button Life	10,000 actuations
Webcam Controls & Effects	LifeCam Dashboard provides access to animated video special effect features and webcam controls. Windows Live Photo Gallery allows you to easily edit and share photos online. Windows Live Movie Maker allows you to start a video project with the click of a button and easily upload your videos. Note: Photo Gallery and Movie Maker integrations are not part of the dashboard itself but part of the actual LifeCam application.
Imaging Features	
Sensor	CMOS sensor technology
Resolution	• Motion Video: 1280 X 720 pixel resolution* • Still Image: 1280 X 800
Imaging Rate	Up to 30 frames per second
Field of View	66°diagonal field of view
Imaging Features	• Digital pan, digital tilt, vertical tilt, and swivel pan, and 4x digital zoom** • Auto focus, range from 6" to infinity • Automatic image adjustment with manual override • 16:9 widescreen • 24-bit color depth
Product Feature Performance	
Audio Features	Integrated microphone and noise cancellation
Microphone Technology	Unidirectional noise cancelling microphone
Frequency Range	Frequency range 200Hz – 7.5kHz
Mounting Features	Flexible universal attachment base
Storage Temperature & Humidity	-40 °F (-40 °C) to 140 °F (60 °C) at <5% to 65% relative humidity (non-condensing)
Operating Temperature & Humidity	32°F (0°C) to 104 °F (40 °C) at <5% to 80% relative humidity (non-con densing)
Certification Information	
Country of Manufacture	People's Republic of China
ISO 9001 Qualified Manufacturer	Yes
ISO 14001 Qualified Manufacturer	Yes
Restriction on Hazardous Substances	This device complies with all applicable worldwide regulations and restrictions including, but not limited to: EU directive 2002/95/EC on the Restriction of the Use of Certain Hazardous Substances in Electrical and Electronic Equipment and EU Registration Evaluation and Authorization of Chemicals (REACH) regulation regarding Substances of Very High Concern.
FCC ID	This device complies with Part 15 of the FCC Rules and Industry Canada ICES-003. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation. Tested to comply with FCC standards. For home and office use. Model number: 1415, LifeCam HD-5000.
Agency and Regulatory Marks	• ACMA Declaration of Conformity (Australia and New Zealand) • ICES-003 report on file (Canada) • EIP Pollution Control Mark, EPUP (China) • CE Declaration of Conformity, Safety and EMC (European Union) • WEEE (European Union) • VCCI Certificate (Japan) • CITC Letter (Kingdom of Saudi Arabia) • KCC Certificate (Korea) • GOST Certificate (Russia) • FCC Declaration of Conformity (USA) • UL and cUL Listed Accessory (USA and Canada) • CB Scheme Certificate (International)
Windows Hardware Quality Labs (WHQL)	ID: 1436293 Microsoft Windows 7

Figure B.1: The specifications of the Microsoft HD-5000 LifeCam webcam. Taken from Microsoft (2015).

Appendix C

Flight Test Images

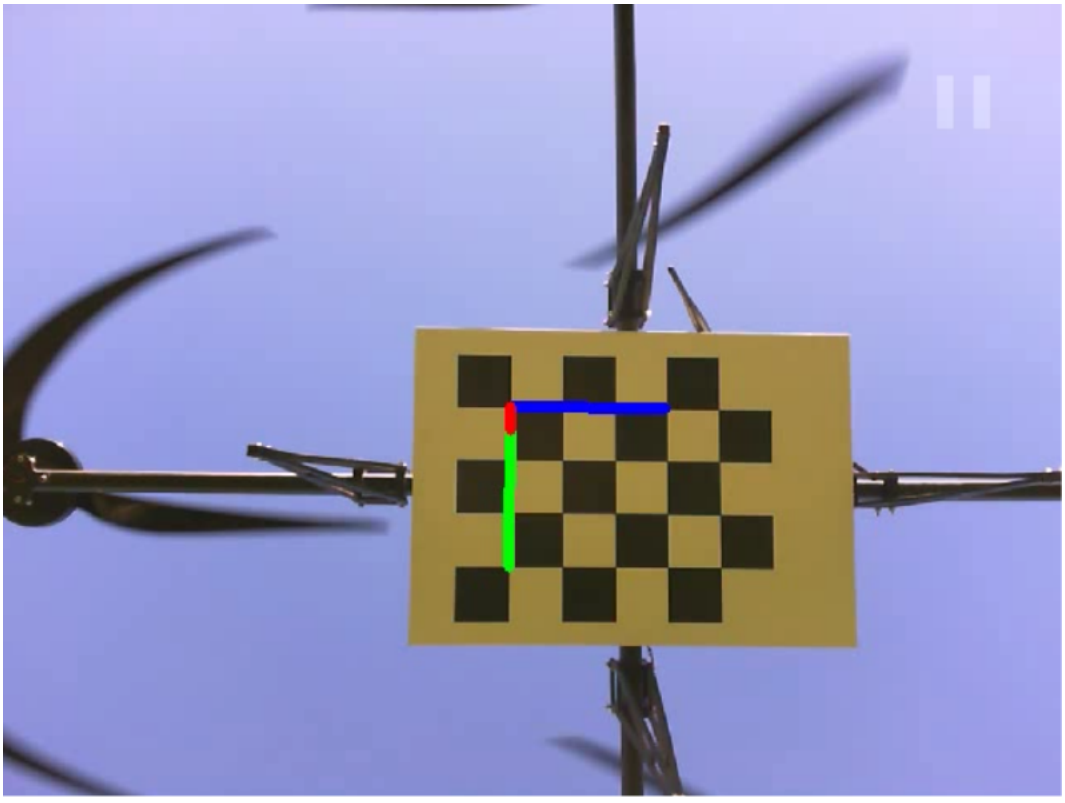


Figure C.1: Picture of a typical test image where the CVS drew an axis system on the board.

Figure C.2: Picture of where the board is partially out of the camera's view. The CVS could not capture all the corner data it required and could not draw an axis system on the board.

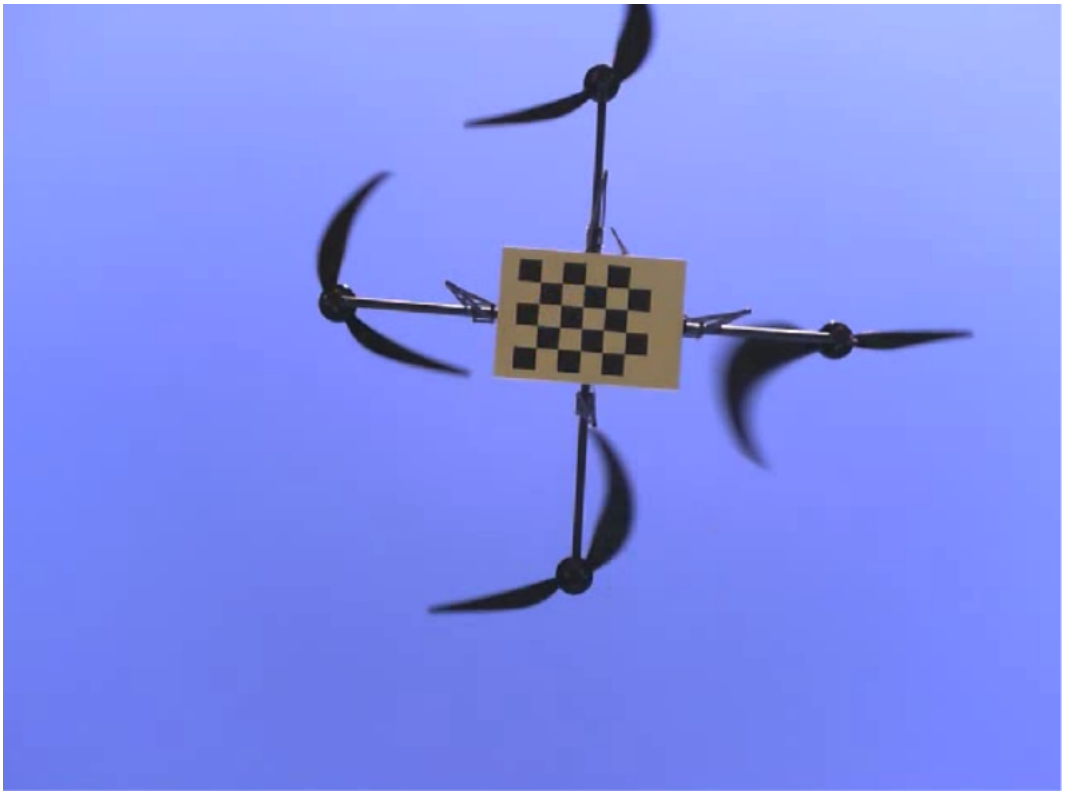


Figure C.3: Picture of where the board is too far away from the camera for the CVS to capture the corner data and draw an axis system.

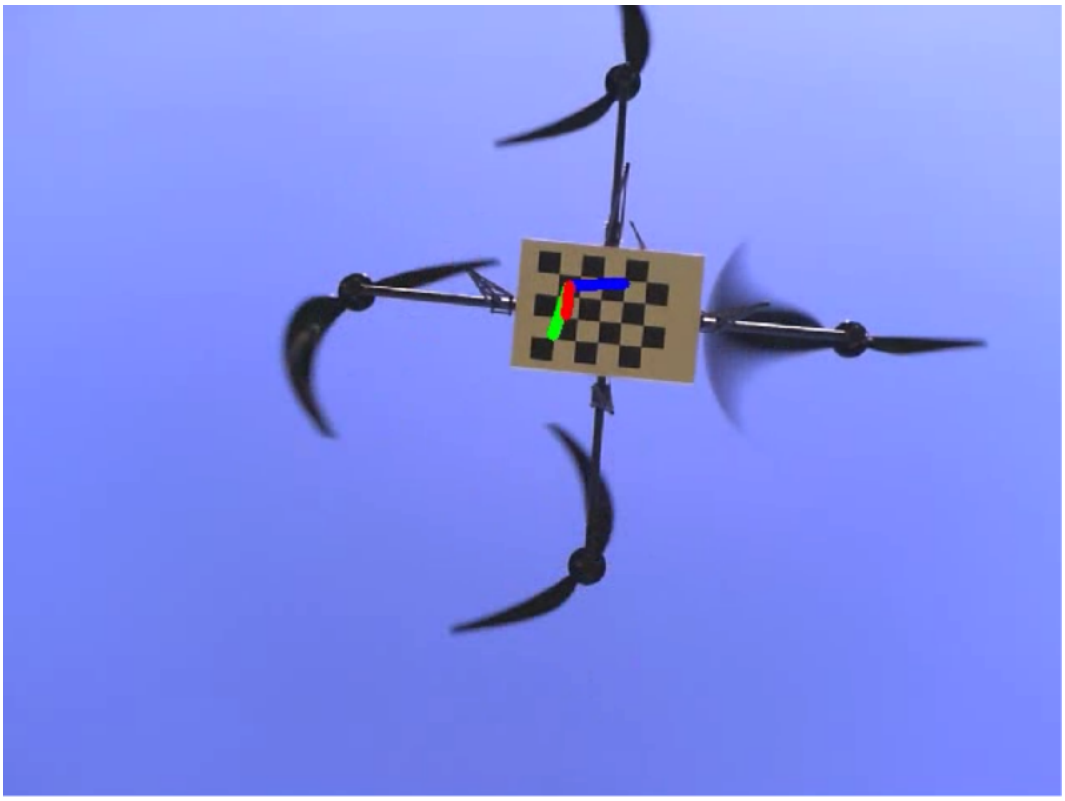


Figure C.4: Picture with an incorrectly drawn axis system. Here, due to the board's distance from the camera and bad lighting conditions, the CVS could not accurately capture the corner information and drew an inaccurate axis system.

Appendix D

Helio100 Site Weather Data

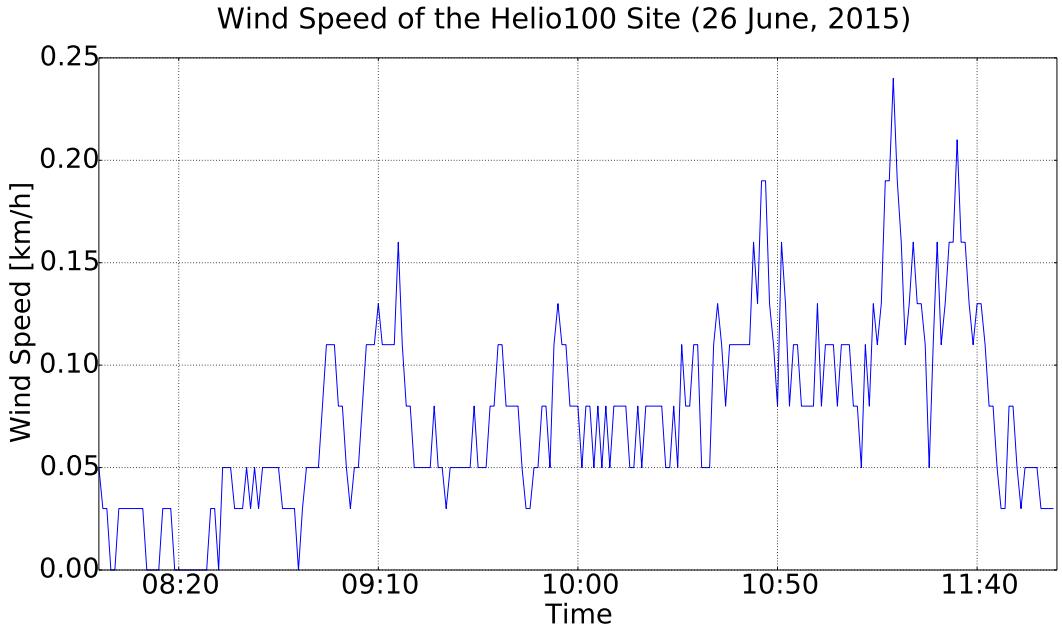


Figure D.1: Wind speed data of the Helio100 site recorded during the flight tests on the 26^{th} of June, 2015.

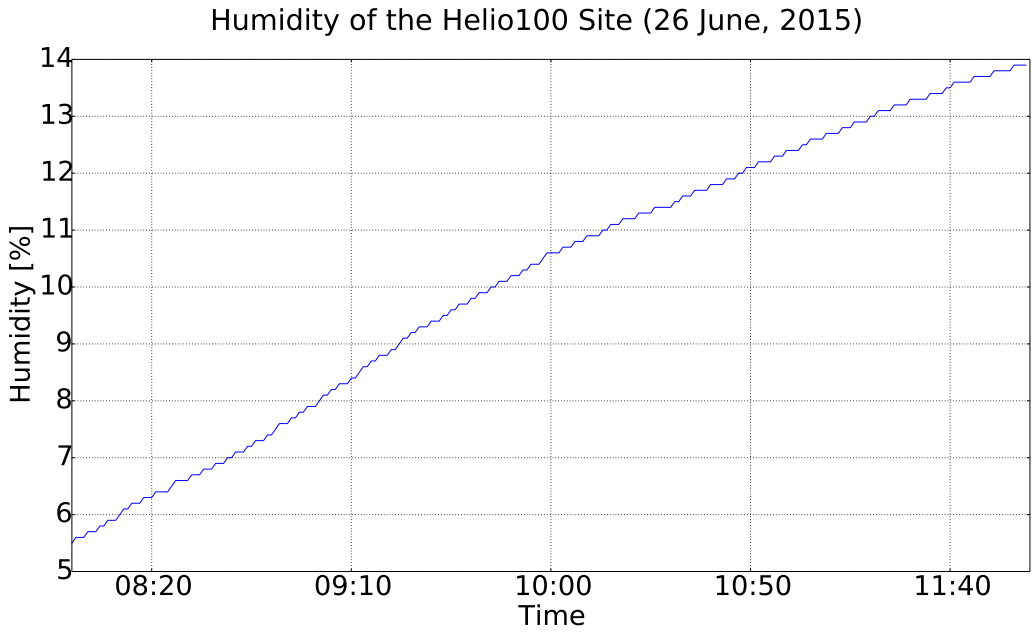


Figure D.2: Humidity data of the Helio100 site recorded during the flight tests on the 26^{th} of June, 2015.

List of References

Amari, S.I. and Wu, S. (1999). Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, vol. 12, no. 6, pp. 783–789.

Arducopter (2015). Pixhawk overview. `bit.ly/1X847Kz`.

Basson, M. (2015 June). Test flight procedure. Personal Communications.

Bouabdallah, S., Noth, A. and Siegwart, R. (2004). PID vs LQ control techniques applied to an indoor micro quadrotor. In: *International Conference on Intelligent Robots and Systems. Proceedings.*, vol. 3, pp. 2451–2456. IEEE.

Bradski, G. (2000). The opencv library. *Doctor Dobbs Journal*, vol. 25, no. 11, pp. 120–126.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, vol. 20, no. 3, pp. 273–297.

Fischler, M.A. and Bolles, R.C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, vol. 24, no. 6, pp. 381–395.

Gao, X.S., Hou, X.R., Tang, J. and Cheng, H.F. (2003). Complete solution classification for the perspective-three-point problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 8, pp. 930–943.

Hamel, T., Mahony, R., Lozano, R. and Ostrowski, J. (2002). Dynamic modelling and configuration stabilization for an x4-flyer. *aa*, vol. 1, no. 2, p. 3.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In: *Alvey vision conference*, vol. 15, p. 50. Citeseer.

Heikkila, J. and Silvén, O. (1997). A four-step camera calibration procedure with implicit image correction. In: *Computer Society Conference on Computer Vision and Pattern Recognition. Proceedings.*, pp. 1106–1112. IEEE.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558.

Horaud, R., Conio, B., Leboulleux, O. and Lacolle, L.B. (1989). An analytic solution for the perspective 4-point problem. In: *Computer Vision and Pattern Recognition, 1989. Proceedings CVPR'89., IEEE Computer Society Conference on*, pp. 500–507. IEEE.

IRENA (2012 June). *Renewable Energy Series: Cost Analysis Series, Volume 1: Power Sector - Concentrating Solar Power*, vol. 2 of *5*. IRENA.

IRENA (2015 January). *RENEWABLE POWER GENERATION COSTS IN 2014*. IRENA.

Jaeger, H. (2001). The 'echo state' approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, p. 34.

Lepetit, V., Moreno-Noguer, F. and Fua, P. (2009). Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, vol. 81, no. 2, pp. 155–166.

Levenberg, K. (1944). A method for the solution of certain non–linear problems in least squares. *Quarterly of Applied Mathemetics*, vol. 2, p. 164.

Lock, J.C., Smit, W.J. and Treurnicht, J. (2015 November). An investigation into multi-dimensional prediction models to estimate the pose error of a quadcopter in a CSP plant setting. Unpublished conference proceedings.

Lu, C.P., Hager, G.D. and Mjolsness, E. (2000). Fast and globally convergent pose estimation from video images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 6, pp. 610–622.

Majumder, A. (2014). How to build a quadcopter-choosing hardware. `bit.ly/1hkLwbp`.

Marquardt, D.W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, vol. 11, no. 2, pp. 431–441.

Medioni, G. and Kang, S.B. (2004). *Emerging Topics in Computer Vision*. Prentice Hall.

Melen, T. (1994). *Geometrical modelling and calibration of video cameras for underwater navigation*. Institutt for Teknisk Kybernetikk, Universitetet i Trondheim, Norges Tekniske Høgskole.

Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (2013). *Machine learning: An artificial intelligence approach*. Springer Science & Business Media.

Microsoft (2015). Lifecam hd-5000. `bit.ly/1I5O4pI`.

Moore, G.E. (1965). Cramming more components onto integrated circuits, electronics, 38: 8 (1965). *online: `http://intel.ly/1K8AwIQ`*, vol. 16.

Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

OpenCV (2015 October). OpenCV Levenberg-Marquardt PnP Optimisation. `bit.ly/1W7kkyZ`.

Pitz-Paal, R. (2005). *ECOSTAR: European Concentrated Solar Thermal Road-Mapping; Roadmap Document (WP 3 Deliverable No. 7)*. DLR.

Pounds, P., Mahony, R. and Corke, P. (2010). Modelling and control of a large quadrotor robot. *Control Engineering Practice*, vol. 18, no. 7, pp. 691–699.

Raffo, G.V., Ortega, M.G. and Rubio, F.R. (2010). An integral predictive/nonlinear Hinf control structure for a quadrotor helicopter. *Automatica*, vol. 46, no. 1, pp. 29–39.

Richards, J.G. (1999). The measurement of human motion: a comparison of commercially available systems. *Human Movement Science*, vol. 18, no. 5, pp. 589–602.

Saxena, A., Chung, S.H. and Ng, A.Y. (2008). 3-d depth reconstruction from a single still image. *International journal of computer vision*, vol. 76, no. 1, pp. 53–69.

Schweighofer, G. and Pinz, A. (2006). Robust pose estimation from a planar target. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 12, pp. 2024–2030.

Skala, V. (2012). Radial basis functions for high dimensional visualization. *VisGra-ICONS 2012*, pp. 218–222.

Slama, C.C., Theurer, C. and Henriksen, S.W. (1980). *Manual of photogrammetry*. Ed. 4. American Society of photogrammetry.

South African Civil Aviation Authority (2015 June). Remotely Piloted Aircraft Systems (RPAS) Regulation - Part 101.

Svozil, D., Kvasnicka, V. and Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, vol. 39, no. 1, pp. 43–62.

Tu, J.V. (1996). Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of clinical epidemiology*, vol. 49, no. 11, pp. 1225–1231.

Turing, A.M. (1950). Computing machinery and intelligence. *Mind*, pp. 433–460.

Tutschku, K. (1995). Recurrent multilayer perceptrons for identification and control: The road to applications. *Univ. Wurzburg, Germany, ser. Research Report Series*.

Wackerly, D., Mendenhall, W. and Scheaffer, R. (2007). *Mathematical statistics with applications*. Cengage Learning.

Wikimedia Commons (2006). Artificial neural networks. `bit.ly/1IYexjC`.

Wikimedia Commons (2011). Kernel machine. `bit.ly/1Pv8esq`.

Wikimedia Commons (2012). Svm separating hyperplanes. `bit.ly/1UO98Di`.

Wilamowski, B.M. and Jaeger, R.C. (1996). Implementation of rbf type networks by mlp networks. In: *Neural Networks, 1996., IEEE International Conference on*, vol. 3, pp. 1670–1675. IEEE.

Windolf, M., Gotzen, N. and Morlock, M. (2008). Systematic accuracy and precision analysis of video motion capturing systems, exemplified on the vicon-460 system. *Journal of biomechanics*, vol. 41, no. 12, pp. 2776–2780.

Xie, T., Yu, H. and Wilamowski, B. (2011). Comparison between traditional neural networks and radial basis function networks. In: *IEEE International Symposium on Industrial Electronics*.